




HP 3000

INTERNATIONAL CONFERENCE
VIENNA 1987

i



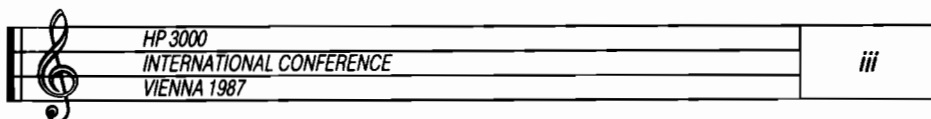
HP 3000
INTERNATIONAL CONFERENCE
VIENNA 1987, 23.-27. MARCH
PROCEEDINGS

	<i>HP 3000</i>	<i>ii</i>
	<i>INTERNATIONAL CONFERENCE</i>	
	<i>VIENNA 1987</i>	

The papers are grouped according to the eight subject categories. Within these categories the order of arrival at the editor's office has been followed. Papers coming in after the copy dead-line are arranged in an own group at the end of the proceedings.

Papers are reproduced exactly as they were submitted. Scientific, grammatical or typographical inaccuracies have not been corrected.

Copyright 1987 by HP3000 ENUG European National User Groups
c/o AUG Austrian Users Group, Vienna



Committees

Organizing Committee

P. Vollsinger (Chairman)
K. Bauer
H. Kässmayer
H. Lenglachner
W. Vitovec
O. Wiegele

PCO

G. Winter
Raiffeisen Reisebüro Ges.m.b.H. Wien
Alserbachstraße 30
A-1091 Vienna
Austria

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Topics of the Conference

The papers are divided into the following subject categories:

AI – Artificial Intelligence and Expert Systems

AL – Advanced Languages

DB – Database Management Systems

This category includes also the papers concerning future DB-systems and the migration to them.

DC – Data Communications and Networks

OA – Office Automation, Networks and PCs

RI – RISC and HP-PA

This chapter contains all papers with the subject “precision architecture” and the migration path from the current systems to the “PA-systems”.

SM – System Management

VS – Various

All other papers which cannot be put in the previous groups are gathered under this mnemonic.

*

– ...

All papers which arrived after the copy dead-line are additionally marked with an “*”. You will find these papers at the end of the proceedings grouped according to the eighth subject categories.

Table of Contents

AI01	An Expert System Manager for the HP3000 Ross G. Hopmans / Brant Computer Services Limited
AI02	A development methodology for expert system programming Shawn Brayman, Ross Hopmans / Brant Computer Services Limited
AI03	Logic Programming and Expert Systems Shawn Brayman / Brant Computer Services Limited
AI04	An Expert Financial Planning System Don MacKenzie, Ross Hopmans, Shawn Brayman / Brant Computer Services Limited
AI05	Artificial Intelligence Applied to the HELP Function Robert Stanley / Cognos Incorporated
AI06	Machine Learning David Price / VRS Software Ltd.
AL01	The analyst workbench revolution Jim Farrow, Stephen Price / DCE Information Consultancy
AL02	System Development and Prototyping Using 4GL's - The Changing Role of the Programmer Jürgen Fritz / JF Management- und DV-Beratung
AL03	Report Generation using a Visual Programming Interface Tim Dudley / Cognos Incorporated
AL04	The Human Interface to 4GL Hans van der Leeuw / Assyst-Raet
AL05	4GL - The Controversy Rages On Karen Heater / Infocenter Ltd.
AL06	Migrating PowerHouse Applications to New Machine Environments Paul Elder, Jim Sinclair / Cognos Incorporated
AL07	Information systems prototyping Orland Larson / Hewlett-Packard
AL08	How to write structured Transact programs Ewald M. Mund
AL09	The 4th generation environment Rudi Huysmans, Wim Bockstaele / Sydes
DB01	Data integrity and recovery Robert Bray / Carolian Systems International Inc.
DB02	Using DBchange to improve your database administrator's productivity Robert Ross / Hewlett-Packard
DB03	Relational Access to IMAGE Data Bases Dr. Wolfgang Matt / Industrieanlagen-Betriebsgesellschaft mbH
DB04	"\$", "*" and Other IMAGE Lists Fred White / Adager
DB05	How to build a distributed M.I.S. system? Roger W. Lawson / Proactive Systems Ltd.
DB06	Normalization - the perfect database? Glen Kalina / Hewlett-Packard



- DB07 A comparison of TurboIMAGE and HPSQL**
Larry Kemp / Hewlett-Packard
- DB08 Relational database: how do you know you need one?**
Orland Larson / Hewlett-Packard
- DB09 Linking to HP System Dictionary**
Ron Harnar / Hewlett-Packard
- DB10 Is there life besides IMAGE?**
May Kovalick / Hewlett-Packard
- DB11 HPSQL in practice**
Andre Van Aken / Hewlett-Packard
- DB12 Maximizing your database through normalization**
Michele Dingerson / Hewlett-Packard
- DB13 Dynamic Aspects of Information Modelling**
Ewald M. Mund
- DB14 Million Records Database Strategies**
Clifford W. Lazar / Systems Express
- DC01 Experience with an IEEE 802.3 Local Area Network in a multi-vendor environment**
Dr. W. Schmatz / MAN Technologie GmbH, München
- DC02 Networking: Will Today's Choices Be Needed Tomorrow?**
Scott Brear / MICOM Systems
- DC03 A uniform interface to distributed services**
Stuart R. Patterson / VICORP Marketing GmbH
- DC04 Network maintenance management**
Brian Button / Hewlett-Packard
- OA01 Making external data available on a micro**
Ronald W. Collison / DARPA
- OA02 Software Super Market / The Gateway project**
Roni Klimscheffskij / Posts and telecommunications of Finland
- OA03 The Mini and the Micro - Distributed Application Development and Processing**
Karen Heater / Infocentre Ltd.
- OA04 PC to HP3000 communications: a perspective**
Sam Patsy / Hewlett-Packard
- OA05 Document filing and retrieval**
Rudi Huysmans, Peter Arfeuille / Sydes
- OA06 Sophisticated Computer-Aided-Publishing in a Commercial Environment**
Dr. Wolfgang Vitovec / Herold
- OA07 Desk Top Publishing - our first three years**
Tim Cullis / HP Computer Users Association UK
- RI01 A Spectrum strategy for the current HP3000 user**
Nick M. Demos / Performance Software Group
- RI02 MPE XL contributions to HP3000 system availability**
Dave Trout / Hewlett-Packard
- RI03 Design features of the MPE XL user interface**
John Korondy, Denis Rachal, Jeff Vance / Hewlett-Packard
- RI04 MPE V to MPE XL migration overview**
R. Gregory Stephens; Speaker: Lee Courtney / Hewlett-Packard
- RI05 Design of the HP3000 Series 950**
Peter Rosenblatt / Hewlett-Packard

- RI06 MPE V to MPE XL migration: migration tools**
R. Gregory Stephens; Speaker: Lee Courtney / Hewlett-Packard
- SM01 Converting to IBM - Are You Sure?**
Donal Barksdale / Spectra-Physics
- SM02 Predicting system performance**
Larry Kemp / Hewlett-Packard
- SM03 Disc performance - what is it?**
Rick Aldinger / Hewlett-Packard
- SM04 Performance measurement for capacity planning**
Tim Twietmeyer / Hewlett-Packard
- SM05 HP Proactive Support Systems, Predictive Support and HPTrend**
Bruce Richards / Hewlett-Packard
- SM06 Is online backup possible outside spectrum**
Joerg Groessler / Joerg Groessler GmbH
- SM07 Application Recovery**
Eduard Stiefel / SWS SoftWare Systems AG
- VS01 „CC - STIPRO“ (Inventory: Stocktaking by Random Samples)**
Josef Angerer / Chemserv Consulting GesmbH
- VS02 The integration of hardware and software maintenance**
Judy Hayner / Hewlett-Packard
- VS03 RINs, RINs, RINs**
Benedict G. Bruno / S.T.R. Software Company
- VS04 The Legal Protection of Software in European and International Law**
Peter R. Ackermann / Orbit Software
- VS05 Comparative Performance of HP3000 Report Writers**
Roger Lawson / Proactive Systems
- VS06 Squeezing the last bit out of your HP3000**
Robert M. Green / Robelle Consulting
- VS07 Migrating Large Software Systems from an HP3000 to a DEC-VAX**
Alexander Kotys / SIS Datenverarbeitung Ges.m.b.H.

The following papers arrived after the copy dead-line. You will find these papers at the end of the proceedings grouped according to the eight subject categories additionally marked with an “*”.

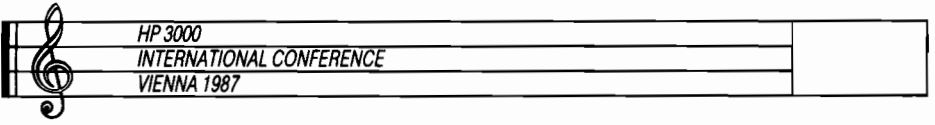
- AL10* Applications design and optimization for the Series /930**
Terri Csete / Hewlett-Packard
- DB15* A friendly system for design and management of IMAGE-databases**
R.Raschetti, B.Caffari
- DB16* SQL and Tools**
Hewlett-Packard Presentation
- DB17* Image migration update**
Hewlett-Packard Presentation
- DB18* Recommendations for defining transactions in Turbo- and HPIMAGE**
Peter Kane / Hewlett-Packard
- DB19* Inter-system database interaction in a network environment**
Dave Mackay / Hewlett-Packard
- DB20* Can Profiler improve your database performance?**
Robert Ross / Hewlett-Packard



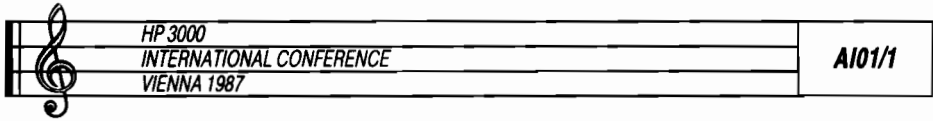
Author's Index

Ackermann, Peter R.	VS04	Kotys, Alexander	VS07
Aken, Andre Van	DB11	Kovalik, May	DB10
Alblas, Henk	SM09	Laidig, Klaus Dieter	VS09
Aldinger, Rick	SM03	Larson, Orland	AL07, DB08
Angerer, Josef	VS01	Lawson, Roger W.	DB05, VS05
Arfeuille, Peter	*OA05	Lazar, Clifford W.	DB14
Barksdale, Donal	SM01	Leeuw, Hans van der	AL04
Bockstaele, Wim	*AL09	MacKenzie, Don	AI04
Bosse, John	SM08	Mackay, Dave	DB19
Bray, Robert	DB01	Matt, Dr. Wolfgang	DB03
Brayman, Shawn	AI02, *AI03, *AI04	McShane, Michael	*DC05
Brear, Scott	DC02	Mund, Ewald M.	AL08, DB13
Bruno, Benedict G.	VS03	Nelson, Deborah	DC06, *DC09
Button, Brian	DC04	Oxford, Richard	DC05
Caffari, B.	*DB15	Patsy, Sam	OA04
Chance, Doug	RI07	Patterson, Stuart R.	DC03
Collison, Ronald R.	OA01	Price, Stephen	*AL01
Courtney, Lee	VS08	Price, David	AI06
Cullis, Tim	OA07	Puette, Bob	OA10
Demos, Nick M.	RI01	Rachal, Denis	*RI03
Dingerson, Michele	DB12	Raschetti, R.	DB15
Dudley, Jim	AL03	Richards, Bruce	SM05
Elder, Paul	AL06	Roelandts, Wim	DC08
Farrow, Jim	AL01	Rosenblatt, Peter	RI05
Fritz, Jürgen	AL02	Ross, Robert	DB02
Green, Robert M.	VS06	Schmatz, Dr. W.	DC01
Groessler, Joerg	SM06	Sinclair, Jim	*AL06
Harnar, Ron	DB09	Spreng, Doug	RI07
Hayner, Judy	VS02	Stanley, Robert	AI05
Heater, Karen	AL05, OA03	Stephens, R. Gregory	RI04, RI06
Hopmans, Ross G.	AI01, *AI02, *AI04	Stiefel, Eduard	SM07
Hornung, Jeff	RI08	Trout, Dave	RI02
Huysmans, Rudi	AL09, OA05	Twietsmeyer, Tim	SM04
Kalina, Glen	DB06	Vance, Jeff	*RI03
Kane, Peter	DB18	Vathauer, Brenda	DC09
Kemp, Larry	DB07, SM02	Vitovec, Dr. Wolfgang	OA06
Klimscheffskij, Roni	OA02	Wallis, Matthew	OA11
Korondy, John	RI03	White, Fred	DB04

In case of the co-author the paper is marked with a "*"!



HP 3000
INTERNATIONAL CONFERENCE
VIENNA 1987



An EXPERT SYSTEM MANAGER for the HP3000

Ross G. Hopmans

Brant Computer Services Limited
6303 Airport Road, Suite 201
Toronto, Canada

(416) 673-9417

ARTIFICIAL INTELLIGENCE IN BUSINESS

The goal of AI scientists has always been to develop computer programs that could in a sense think; that is, solve problems in a way that would be considered intelligent if done by a human.

To act intelligently - to think - means more than to calculate or match numbers or letters. To think means, among other things, to combine new information, previously known facts, rules of thumb, guesses and even intuition to come up with an appropriate response to a problem.

Artificial Intelligence is a field of study that involves computer program and mechanisms that emulate some aspect of intelligent human behaviour. AI includes such areas as:

- . Expert Systems
- . Robotics
- . Vision Recognition
- . Voice Recognition

The AI marketplace is projected to be the fastest growing sector of the computer industry with general purpose computers the medium of choice.

Brant Computer Services has brought MPROLOG to the HP3000. MPROLOG is an ideal tool for developing prototype and production AI applications. With its external language interface, users of the HP3000 may find that the real power of MPROLOG is its ability to provide "knowledge processing" to commercial applications and data bases.

Knowledge processing systems may be used to solve small, specific problems, supplant training, and enable employees to perform tasks that previously required access to senior specialists.

Most observers agree that software is steadily becoming increasingly complicated, powerful and intelligent. The development of AI applications often requires capabilities not available or feasible using conventional computer languages. The MProlog language provides the technology for developing and executing intelligent and sophisticated applications.

MProlog's use need not be restricted to problems requiring the direct employment of AI technology, however. It has been designed to foster improvements in the conventional software development market. Applications which may not require AI techniques can be implemented and maintained more productively by employing AI tools that can intelligently help locate and diagnose errors in the program code.

"Expert Systems" use artificial intelligence techniques to draw on a base of knowledge supplied by a company's experts, including subjective judgments to simulate human reasoning. The systems can make reasoned assumptions to fill in missing information. In contrast, most conventional data processing programs do little more than crunch numbers.

This paper discusses the development of a new product for the HP3000 and PC compatible micro-computers - An Expert System Manager for the HP3000. This product is a knowledge system which uses AI techniques to perform some of the advisory functions of a system manager for the Hewlett-Packard HP3000 line of business computers.

EXPERT SYSTEMS

Among the most significant of the developments in the field of Artificial Intelligence are "expert" or "knowledge-based" systems. These programs are designed to represent and apply factual knowledge of specific areas of expertise to solve problems.

Like human experts, these systems use symbolic logic and heuristics - rules of thumb - to find solutions. Expert systems may make errors from a lack of knowledge but should be able to learn from their mistakes and, unlike their human counterparts, provide a permanent, consistent record of the knowledge they contain.

Initially, AI scientists tried to simulate the complicated process of thinking by finding general methods for solving broad classes of problems. However, the more classes of problems a single program could handle, the more poorly it seemed to do on any individual problem. They next decided to concentrate on developing general methods or techniques to use in more specialized programs. It wasn't until the late seventies that AI scientists began to realize

that the problem-solving power of a program comes from the knowledge it possesses, not just the formalisms and inference schemes it employs. In other words, to make a program intelligent, provide it with lots of high-quality, specific knowledge about some problem area.

This realization led to the development of special-purpose computer programs that were expert in some narrow problem area. These programs were called expert systems.

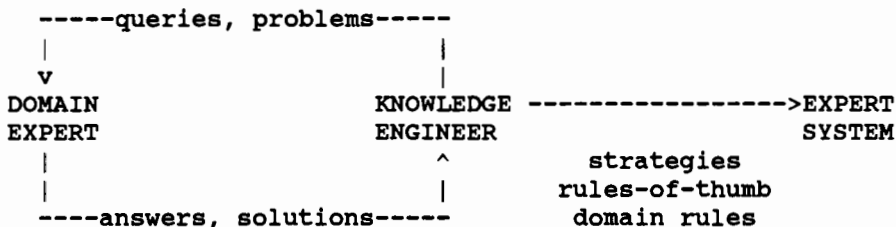
The process of building an expert system is often called Knowledge Engineering. It typically involves a special form of interaction between the expert-system builder, called the knowledge engineer, and one or more experts in some problem area. The knowledge engineer "extracts" from the human experts their procedures, strategies, and rules of thumb for problem solving, and builds this knowledge into the expert system.

The result is a computer program that solves problems in much the same manner as the human experts. An expert described by Paul E. Johnson is a person who, because of training and experience, is able to do things the rest of us cannot; experts are not only proficient but also smooth and efficient in the actions they take. Experts know a great many things and have tricks and caveats for applying what they know to problems and tasks; they are also good at plowing through irrelevant information in order to get at basic issues, and they are good at recognizing problems they face as instances of types with which they are familiar. Underlying the behaviour of experts is the body of operative knowledge we have termed expertise. It is reasonable to suppose, therefore, that experts are the ones to ask when we wish to represent the expertise that makes their behaviour possible.

Knowledge engineering relies heavily on the study of human experts in order to develop intelligent, skilled programs. As Hayes-Roth and others point out in the book Building Expert Systems, the central notion of intelligent problem-solving is that a system must construct its solution selectively and efficiently from a space of alternatives. When resource-limited, the expert needs to search this space selectively, with as little unfruitful activity as possible. An expert's knowledge helps spot useful data early, suggests promising ways to exploit them, and helps avoid low-payoff efforts by pruning blind alleys as early as possible. An expert system achieves high performance by using knowledge to

make the best use of its time.

The transfer of knowledge from an expert to a computer program (knowledge engineering) can be pictured as follows:



The main players in the expert system game are the expert system, the domain expert, the knowledge engineer, the expert-system-building tool and the user.

The expert system is the collection of programs or computer software that solves problems in the domain of interest. It is called a system rather than just a program because it contains both a problem solving component and a support component. This support environment helps the user interact with the main program and may include sophisticated debugging aids to help the expert-system builder test and evaluate the program's code, editing facilities and advanced graphics.

The domain expert or area expert is an articulate, knowledgeable person with a reputation for producing good solutions to problems in a particular field. The expert uses tricks and shortcuts to make the search for a solution more efficient, and the expert system models these problem-solving strategies. Although an expert system usually models one or more experts, it may contain expertise from other sources such as books or journal articles.

The knowledge engineer is a human, usually with a background in computer science and AI, who knows how to build expert systems. The knowledge engineer interviews the experts, organizes the knowledge, decides how it should be represented in the expert system, and may help programmers to write the code.

The expert-system-building tool is the programming language used by the knowledge engineer or programmer to

build the expert system. These tools differ from conventional programming languages in that they provide convenient ways to represent complex, high-level concepts. The term tool usually refers to both the programming language and to the support environment used to build the expert system.

The user is the human who uses the expert system once it is developed.

Why develop expert systems? One advantage is permanence. Human expertise can fade quickly whereas artificial expertise is around forever. Another advantage is the ease with which it can be transferred or reproduced - not true for humans. Artificial expertise is easy to document and produces more consistent, reproducible results than does human expertise. Computer programs are not susceptible to time pressures, stress or distractions. Finally, human experts are very scarce. Expert systems allow the expertise to be readily attainable and relatively inexpensive.

THE STRUCTURE OF AN EXPERT SYSTEM

We use the term knowledge to mean the information a computer program needs before it can behave intelligently. This information can take the form of facts or rules.

Facts and rules in an expert system are not always either true or false; sometimes there is a degree of uncertainty about the validity of a fact or the accuracy of a rule. When this doubt is made explicit, it is called a certainty factor.

Many of the rules in expert systems are heuristics - rules of thumb or simplifications that effectively limit the search for solutions. Expert systems use heuristics because the tasks these systems undertake are typically difficult and often poorly understood. They tend to resist rigorous mathematical analysis or algorithmic solutions.

The knowledge in an expert system is organized in a way that separates the knowledge about the problem domain from the system's other knowledge, such as general knowledge about how to solve problems or knowledge about how to interact with the user. This collection of domain knowledge is called the



knowledge base, while the general problem-solving knowledge is called the inference engine. A program with knowledge organized this way is called a knowledge-based system.

The knowledge base in an expert system contains facts (data) and rules that use those facts as the basis for decision making. The inference engine contains an interpreter that decides how to apply the rules to infer new knowledge and a scheduler that decides the order in which the rules should be applied.


How the system uses its knowledge is important. An expert system must have both the appropriate knowledge and the means to use the knowledge effectively to be considered skilled at some task. Thus to be skilled, an expert system must have a knowledge base containing lots of high-powered knowledge about the problem domain and an inference engine containing knowledge about how to make effective use of the domain knowledge.

Finally, we need to consider how the knowledge is represented. Rule-based representations centre on the use of IF condition THEN action statements. Frame-based knowledge representation uses a network of nodes connected by relations and organized into a hierarchy.

Expert systems manipulate knowledge while conventional programs manipulate data. Teknowledge characterizes the differences between expert systems and ordinary programs as follows:

Data Processing	Knowledge Engineering
. Representations and use of data	. Representations and use of knowledge
. Algorithmic	. Heuristic
. Repetitive process	. Inferential process
. Effective manipulation of large data bases	. Effective manipulation of large knowledge bases

When human experts solve problems, particularly the type we consider appropriate for expert system work, they chose symbols to represent the problem concepts and apply various strategies and heuristics to manipulate these concepts. An expert system also represents knowledge symbolically. The symbols can be combined to express relationships between

	HP 3000	A101/8
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

them. To solve a problem, an expert system manipulates these symbols and the consequence is that the knowledge representation becomes very important.


An expert system has depth; that is, it operates effectively in a narrow domain containing difficult, challenging problems. Thus the rules in an expert system are necessarily complicated through their individual complexity or sheer number.

An expert system has knowledge that lets it reason about its own operation plus a structure that simplifies this reasoning process. For example, if an expert system is organized as sets of rules, then it can easily look at the inference chains it produces to reach a conclusion. If given rules that tell it what to do with these inference chains, it can use them to check the accuracy, consistency, and plausibility of its conclusions and justify its reasoning. Most current systems have an explanation facility to explain how the system arrived at its answers and these usually involve displaying the inference chains and explaining the rationale behind each rule used in the chain. This provides a self-knowledge which is becoming more important in helping expert systems to resolve inconsistencies.

MPROLOG - THE DEVELOPMENT TOOL

MPROLOG is an advanced, modular implementation of the Prolog language. Selected as the basis for the Japanese Fifth Generation Computer System Project, Prolog enables designers to describe their application in logical terms for interpretation by the computer. It will help solve problems involving all types of logical reasoning quickly and easily.

MPROLOG gives you a powerful inference engine surpassing the capabilities of most expert shells. MPROLOG provides automatic, system-driven reasoning with the rules and facts in the program knowledge base. You describe the problem by describing the relevant properties of the objects you are interested in, and the relationships between different objects. The MPROLOG uses its build-in logical inference engine to find a solution to your problem. By way of contrast, to solve a problem with a traditional programming language, you must give precise, step-by-step instructions for how that problem is to be solved.

	HP 3000	AI01/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

In addition, MPROLOG gives you interfaces to procedural languages, hardware and operating system independence, high performance and a program development environment with on-line help, interactive editing, error correction and program trace facilities.

MPROLOG operates in a number of computer environments in addition to the HP3000 including IBM VM/CMS, IBM MVS/TSO, IBM PC-DOS, DEC VAX/VMS, DEC VAX/UNIX, CDC Cyber, and AI workstations including Tecktronix 4404, Sun, Apollo, Charles River, Mostek and other M68000/UNIX.

Such features as floating point arithmetic, exception handling, dynamic programming, type checking, host and external language interfaces make MPROLOG a robust tool for the development of application programs. Operating system independence means that only a single source must be maintained for applications running in different production environments.

The programming and production environment consists of the following five components:

The Program Development Support System (PDSS)
for writing, editing and testing MProlog programs


The Pretranslator
translates MProlog source modules into binary form

The Compiler
compiles these binary modules into machine-code

The Consolidator
links pretranslated and compiled modules together,
forming a standalone MProlog program

The Interpreter
executes a standalone MProlog program

Brant Computer Services has brought MPROLOG to the HP3000. MPROLOG has a high profile in a relatively sound industry and Brant has exclusive, world-wide rights to market the language to all Hewlett-Packard computer systems.

	HP 3000	AI01/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

THE APPLICATION - AN EXPERT SYSTEM MANAGER

Brant is a supplier of computer software, services and support. The Expert System Manager was conceived for two reasons - to supplement our Facilities Support offering and as a documentation and educational tool to help us capture the knowledge of our in-house experts and to help train others.

The Expert System Manager was designed with an unsophisticated end-user in mind. The product must run on the micro-computer in case the HP3000 goes down and must converse with the user in a question and answer fashion. The system should lead to specific recommendations to remedy the problem with a minimum number of questions and should help the user restore the computer to an operational state as quickly as possible. The system should, in addition, explain its reasoning, address preventative maintenance and allow users to add to the base of knowledge in the system.

Rather than a strictly academic exercise, this product was conceived not only as saleable, but was seen as required in many situations. The Expert System Manager will be quite useful to new uses of smaller systems who do not wish to hire specific data processing staff.

Large shops can also benefit from the Expert System Manager to overcome problems which occur when the system manager is unavailable. Multiple site shops can use the Expert System Manager to maintain consistency across machines and sites. Since the base of knowledge is expandable, it can be tailored to the specific requirements and methodology where it is installed.

It is important in the development of any expert system to restrict its scope to a manageable size. System managers are responsible for solving numerous and varied problems which are often site and application specific.

We decided to limit the scope of the project to the set of problems concerned with the operating system and communications from the end-user to the host computer. Our problem profile is quite broad, covering numerous problems that end-users and computer operators encounter in trying to establish communications with the operating system and low-level problems that they may experience once they have the attention of the operating system.

The purpose of the system is, through communication with the end-user, to establish the nature of the problem and recommend a means of overcoming it. If a solution suggested in not successful or not available, the system will suggest that the user seek outside help and prompt for the solution to be added to its knowledge base.

A SYSTEM MANAGER'S ROLE

System managers are responsible for solving numerous and varied problems, as we learned in the observation of our expert. The system manager has prime responsibility for the HP3000 and peripherals. This role extends beyond the machine itself and includes communications to direct and remote users, the operating system, system software and application programs.

The system manager is generally the most knowledgeable person in all aspects of the real world which interact physically in some way with the computer system.

The types of problems which system managers are responsible for solving include:

- . Operating System Problems

The system manager must keep current with known problems in the operating system, the use of the operating system including utilities from HP, the contributed library or those internally developed, keeping the system running, interpreting error messages and how to recover if the system goes down.

- . Operations Problems

Some system managers fill the dual role of computer operator as well, responsible for the console commands including fences and limits, console replies, tape and form mounts, backup, cold load and reload procedures, system loading and performance monitoring

. System Configuration Problems

Configuration of the system is an on-going concern to system managers. System table sizes must be optimized for the user base and the configuration must be kept current with the changing needs of the user base.

. Performance Enhancement Problems

The system manager may be the first to know if a new application is experiencing performance problems through the other users on the system. His or her knowledge must cover data base design, file access techniques, program segmentation and much more.

. Communications Problems

Communications problems can include problems with the terminal, power supply, cabling or configuration of the users, the configuration of the system, modem, multiplexer or line problems with remote users and with application software. The perceived problem may simply be the result of a slow system, incorrect use of application software or may involve local printers or PCs.

These are only some of the problems which system managers may encounter in a typical day on the job. Pressures from end-users, management and the machine itself make the system manager's a very responsible role.

The job of system management, however, is often very site specific. Experienced system managers are in short supply. After hiring one, he must still be trained in the inner workings of the new site and will certainly not be available for the twenty-four hours a day, seven days a week that the machine is working.

An obvious need exists, then, for an advisory tool to help less highly trained staff solve some of the problems they may encounter, to help these people further qualify difficult problems before calling in the system manager, to document the expertise of the system manager and to help train others in this expertise.

The Expert System Manager project was undertaken to fulfill this need.



THE KNOWLEDGE ENGINEERING

To obtain the knowledge base necessary for our Expert System Manager, we used two of our system managers as the experts along with the HP manuals and trade journals.

A paradox of expertise is that the more competent domain experts become, the less they are able to describe the knowledge they use to solve problems. Domain experts need outside help to clarify their thinking and that is the role of the knowledge engineer.

Our goal in the knowledge acquisition was to transfer the problem solving expertise from our human system managers to a computer program.

The knowledge engineer was not himself an expert in system management or operations. He obtained most of his information through direct interaction with each of the system managers independently.

Our knowledge engineering sessions took three basic forms. One form was through question and answer sessions during which the expert would explain what types of problems he encountered, how he solved the problem and why he took the direction or made the decisions he did. The knowledge engineer would ask for clarification where required and challenge the expert when his reasoning seemed contradictory.

The second form of the knowledge engineering was through case studies. The knowledge engineer presented the expert with realistic or actually documented problems to solve - providing the expert with the actual end-user view of the problem and symptoms as reported. The expert would then proceed to solve the problem as best he could. He would query the knowledge engineer as he would the end-user and conclude what actions he would probably take. The knowledge engineer took an active role, requesting explanations from the expert at intervals in the process.

The third form of the knowledge engineering took the form of strict observation during the expert's actual problem solving process. The knowledge engineer spent time with the expert on the job and recorded the actions taken and questions posed by the expert in solving problems as they arose.


No matter what the form of the knowledge engineering, our knowledge engineer was keen to note the organizational mechanisms used by the expert to classify the type of problem he encountered. The organizational constructs formed the basis for certain types of inferences the expert makes during problem-solving and constitutes the structural expertise about the domain.

The knowledge engineer also listened for the basic strategies the human experts used when performing the task of problem-solving. What facts did they try to establish first? What kinds of questions did the experts ask first? Did the experts make initial guesses about anything based on tentative information? How did the human experts try to refine their guesses? In what order did they pursue each of the subtasks and how did this order vary in case studies? How did the expert justify his problem-solving techniques?

Given any particular situation, it was important to identify the actual problem, its characteristics and sub-problems. The objective was to characterize the supporting knowledge structure for each problem to begin building the knowledge base. Several iterations were required for each problem. Some problems were split into multiple problems if considered too large. The following points were considered important:

- . what classes of problems does the expert solve?
- . how are the problems defined?
- . how are the problems partitioned?
- . what are the important data?
- . what situations are likely to impede solutions?
- . what does a solution look like?

It was important to us not to ask our experts directly about their rules or methods for solving a problem. In general, domain experts have great difficulty in expressing such rules. They have a tendency to state their conclusions and reasoning in general terms, too broadly for effective machine analysis. The human expert makes complex judgments rapidly. The pieces of basic knowledge are assumed and combined so quickly that it is difficult for the expert to

	HP 3000	AI01/15
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

describe the process. He may be even be unaware of the individual steps taken to reach a solution. He may call it intuition or a hunch - but he uses a complex reasoning process based upon a large amount of remembered data and experience. Interestingly enough, studies have shown that when experts attempt to explain how they reached a conclusion, they often construct plausible lines of reasoning that bear little resemblance to their actual problem-solving activity.

Hence, we did not believe anything our experts said outright. Working hypotheses were developed based on information from the expert which was tested for validity and consistency before being accepted. The tests involved having the expert solve new problems using the hypothesis - the experts had to demonstrate the use of their rules during actual problem solving.

Within their area of expertise, our experts quickly recognized new situations as instances of things with which they were already familiar. However, when faced with new situations, they applied general principles and deductive steps. It was more insightful to us to present the experts with novel problems to decompile their knowledge and view the actual problem solving activity.

DEVELOPMENT OF THE PROTOTYPE

Building an expert system - like any major systems development exercise - can be a monumental task. However, like any application development project, prototyping techniques can be employed to quickly build a working model of the final expert system. The prototype can incrementally evolve into a complete system, effectively solving the problem of system definition.

For any knowledge system, the issues are the knowledge engineering, the knowledge representation and the user interface. As our knowledge engineering progressed, the knowledge representation evolved as a prototype was developed. Both the knowledge representation and user interface could be changed as required to suite the evolving needs.

We decided to implement an explanation facility into the model. The program poses questions to the user who answers with "YES", "NO" or "WHY". The program decides what question to ask next based on answers to previous questions. If the user asks "WHY", the program will explain why the question is being asked, in the context of the user's answers to previous questions.

The program must first ascertain that there is a problem with the computer system. Hence, the first question the user encounters is:

Is there a problem with the computer system? ==> YES
 NO
 WHY

We have partitioned the domain into three areas: the operating system, communications and applications. The user is asked if there appears to be a problem in any of these areas, each subsequent question driven by a "NO" to the previous area as follows:

Is there a problem with the operating system? ==> YES
 NO
 WHY

Is there a communications problem? ==> YES
 NO
 WHY

Is there a problem with an application? ==> YES
 NO
 WHY

If we take communications, for example, we try to ascertain the nature of the problem through the use of questions such as:

Does a cursor appear on the screen? ==> YES
 NO
 WHY

Is yours the only terminal with a problem? ==> YES
 NO
 WHY

Can you type any characters onto the screen? ==> YES
 NO
 WHY



We go on to give specific instructions about how to go into local or remote mode or check the configuration of the terminal as required. If the user asks "WHY", the program will explain the purpose of the test or the rule which caused the question to be asked.

If none of the questions seems to be leading to a specific problem area in, say, communications, then the user will be led into the operating system section. Otherwise, specific tests will be suggested and the results queried. If all the tests are applied and the problem persists, the user is advised to call in the actual system manager and document the problem, symptoms and solution.


THE PRODUCT

As the prototype evolves into a product, there are a number of areas which we must build in to be truly usable.

The first of these is the concept of payback versus ease of performing the test. If for example, we appear to have a "hung" terminal, we know that we can clear it by taking the system down and performing a warm start. In an isolated environment this may be perfectly adequate. In most environments, however, it is a poor solution. We must pursue a line of questioning which may get quite involved and the likelihood of success on each individual test may be quite remote. Hence, we need to incorporate information about the environment in which the Expert System Manager is used, the sophistication of the end-user and the relative importance of the situation at hand before we can build a universal product.

We have not, as yet built the ability for the end-user to add to the knowledge base into the product. Although this is relatively easy to implement, the implications of adding potentially conflicting facts and rules into the knowledge base are immense. We do recognize that the ability to add site specific knowledge is critical and plan to implement that facility with sufficient security.

The Expert System Manager contains reasonable generic problem-solving expertise at this point. However, as the product gains sophistication, we will have to add operating

	HP 3000	AI01/18
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

system and device specific knowledge to answer the increased user demand. Once the precision architecture machines are released, for example, users may be faced with a completely new set of problems which the Expert System Manager should be able cope with.

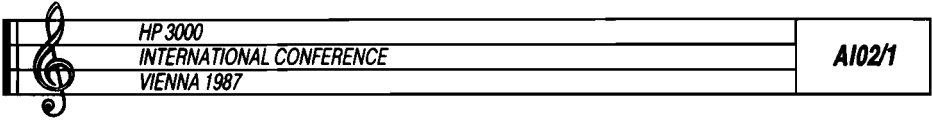
CONCLUSION

Like any new product, the Expert System Manager fills a need and creates a new need. We will be resolving the issues addressed by this project to bring the product to market.

What we have done is use Artificial Intelligence to solve a practical problem. The problem is the scarcity of applied expertise in a particular domain and the solution uses MProlog to capture the knowledge of an expert in that domain to make it commercially available.

Knowledge systems can be developed as prototypes in a matter of months to prove the viability of AI solutions. Knowledge or expertise can be continually added to the system to increase its "intelligence" with no change to the program itself. Knowledge systems can be developed as standalone programs like the Expert System Manager, or they can be developed as an adjunct to existing commercial applications - sharing files and data bases to incorporate expertise into data processing

February 1, 1987.




A DEVELOPMENT METHODOLOGY FOR EXPERT SYSTEM PROGRAMMING

Shawn Brayman and Ross Hopmans

Brant Computer Services Limited
6303 Airport Road, Suite 201
Mississauga, Ontario
Canada

(416) 673-9417

	HP 3000	A102/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

1.0 Introduction

It is necessary in any software development project to have a pre-defined methodology established that allows the developer and management to measure the progress and success of a development effort.

Over the past 18 months Brant Computer Services has undertaken three expert system development projects - one for a client and two internal projects - to develop products. All of our development work has been in the MPROLOG language, a modular implementation of PROLOG that is widely installed on a number of different computer architectures. The development work has been done on the HP Vectra (an IBM compatible) and the source code is compatible with the implementation of MPROLOG for the HP3000 which Brant is currently completing.

The first system Brant began working on was in late 1985 for a life insurance company in Canada. The system was a personal financial planning system and involved a standard 4GL component and the MPROLOG based advisor. Brant worked with one of the top financial planners in Canada on the knowledge engineering phase, and since that time has entered into an agreement to continue the development of the system and market it as a product on the HP3000 and IBM-PCs.

The second system was an internal project from the start and involved the development of a diagnostic expert system for the "system management" of an HP3000 facility. Our first iteration involved the development of a simple expert system shell which we subsequently found incapable of handling the representation of the problem. We are currently developing a more specific diagnostic shell which we will then populate with the "knowledge" of the HP3000.

The third project was the development of a prototype system for a California based agricultural management company. The system is designed to schedule irrigation events for farmers growing specific crops. The system takes into account climatic information, knowledge about the type of crop, specific information on the fields (soil type, salinity etc.) and management constraints to schedule a season's irrigation requirements.

Through our experience in the development of these three systems, coupled with our studies of literature on AI development methodologies, we are developing a set of guidelines to be followed by our staff to help introduce them to this new area of endeavour.

In the body of this paper we will first review some general observations about the knowledge engineering process and the things to look for in selecting a project that you intend to address with a tool like MPROLOG. After this we will outline in more general terms some of the major aspects in the development of an expert system or knowledge-based system.

For people who are not yet familiar with many of the concepts of AI or Logic Programming we would suggest that you read the paper on "Logic Programming and Expert Systems" which is an introductory paper, also in these proceedings.

2.0 Knowledge Engineering

A critical aspect of developing an expert system is the process of knowledge engineering. A knowledge engineer plays the same role as a traditional systems analyst, with a few important differences.

A knowledge engineer is an individual who attempts to discover the rules or heuristics that a human expert uses to reach conclusions in problem solving. A traditional systems analyst may ask a user specific questions about an application and hope to get a reasonable answer. Knowledge engineers are warned on the other hand not to expect a reasonable answer and in fact not to take what the expert says at face value.

Research has uncovered a phenomenon known as the "Paradox of Expertise" - the more competent domain experts become, the less able they₁ are to describe the knowledge they use to solve problems!

It appears that experts use what can be called "compiled expertise", where they may take dozens of specific rules or conditions and compile them all down to one specific rule. It is the knowledge engineer's task to "decompile" this

expertise to a form that is usable in our automated systems. The problem is that when asked, your expert will not recognize that some of his "rules" are actually compilations and so will provide the knowledge engineers with inaccurate or misleading information. In other words the expert may make up false rules to try and explain why he or she made a decision, because they do not know themselves.

A spin-off recommendation that quickly becomes evident from this "Paradox of Expertise" is that you should never be your own expert in a knowledge engineering exercise. The process of "decompiling" the knowledge is not a process that can be effectively undertaken by one individual on his own.

There are two main approaches to knowledge engineering; the observational and the intuitive method:

Investigators applying the observational method don't interrupt the expert with questions or comments during problem solving. Instead, they analyze a transcript of the session after the fact, possibly with the expert's help. AI researchers have used this approach to study problem solving by nonexperts, calling it protocol analysis. Here the subject talks while solving a simple problem or puzzle, the verbalizations are transcribed, and the underlying problem-solving processes are inferred from the resulting trace.

In the case of the intuitive method the knowledge engineer gains a much deeper understanding of the problem domain through readings in the field and interaction with experts. The knowledge engineer then develops a set of rules that he feels adequately represents the "knowledge" and verifies these against the opinions or judgments of the experts. A reversal of this process is when the expert tries to develop the rules himself and provides them to the knowledge engineer. The knowledge engineer then enters them into the system and validates their effectiveness through other experts and the success of the system.

In most real world cases the knowledge engineering process is a combination of both the observational and the intuitive approach.

3.0 Selecting the Appropriate Problem

Not all problems lend themselves to expert system technology, and a number of pitfalls and observations have been accumulated in almost every book on the subject. In this section we will outline a number of the most often quoted reasons for selecting a problem, along with some observations on potential pitfalls.

Traditional Technology doesn't work

If you can solve the problem in a reasonably cost effective manner using traditional languages like "C" or "Fortran" - do it! Do not waste your time and money making an AI tool do what can already be done most effectively with a traditional language. The converse of this is a problem that many people get into where they decide to develop in a traditional language because of portability or performance. Develop your system in a high-level expert system tool so that the solution can be demonstrated in a reasonable length of time. One complete, you can consider re-implementing in a traditional language if required.

Narrow domain of application

For a successful expert system, it is important that the domain of application be as narrow as possible. For this reason most systems work in domains where there is a finite number of rules and relationships to ensure that the solution does not rely on facts that lie outside of its knowledge base. A knowledge base on fault diagnosis for a disk drive on a computer system is limited but detailed enough to warrant the use of an expert system. On the other extreme, an automated psychotherapist would be an extremely bad choice for an expert system as it would need to know a little about the whole world to be effective. The more constrained the knowledge domain, the more successful the project.

There are recognized experts

The experts in a field must be able to perform reasonably well in solving the type of problems being considered. If they do not, replicating poor performance on a computer may not prove fruitful. The experts should use heuristics or

"rules of thumb" in solving the problems and will probably have a vocabulary of abstract terms and concepts. The experts must be better than amateurs.

The task is primarily cognitive

Problems that are undertaken should be cognitive in nature as opposed to numeric, leading to non-numeric solutions.

The task is combinatoric

The task should have lots of choices to make it interesting and to take advantage of the descriptive as opposed to procedural nature of the program. The process of arriving at a solution should involve a chain of reasoning where one rule or fact leads to the firing of a second rule or fact and so on. If the process is actually a shallow line of reasoning on a large number of related but unconnected factors then AI technology does not add a lot to the process.

The problem is at the right stage of knowledge formalization

If an algorithm exists for the solving of a problem, then it is not necessary to apply an AI methodology. If on the other extreme no rules at all apply, then there will be no basis on which the problem can be solved. AI can be considered if the use of algorithms are too slow (ie: some scheduling or planning systems) or if they do not provide sufficient information.

Specialists agree on the knowledge

If there is no agreement on the underlying knowledge base and rules that the system uses, then the results of the process will be of little value as no one will accept the recommendations except that group of specialists who input the rules in the first place.

Other criteria that should be considered in the selection of a problem are:

- The skill is routinely taught to neophytes
- Data and case studies are available
- Incremental progress is possible

- Not a time critical application
- Freedom to fail
- Resulting system would have a high payoff
- Payoff is easily measurable
- Potential users are enthusiastic.


It should be recognized that many problems may meet these criteria without being a full-fledged expert system development project. These smaller systems (better referred to as knowledge-based systems) reflect many of the strengths of the technology without having the same level of investment by the company. They can prove to be very adequate learning exercises with a valuable system as the result, and set the stage for larger projects down the line.

4.0 A General Design Methodology

Up until recently, the knowledge engineer was himself an expert and an expert in short supply. The actual term was coined at a point in time when the process of refining the knowledge of a domain expert was of such a magnitude that the term "engineer" seemed appropriate. Many of the observations that have led to the outline of design methodologies for expert systems are themselves the results of refining the years of experience, the heuristics, of knowledge engineers.

... acquiring knowledge from experts resists linear, one-pass techniques. Instead, knowledge acquisition and system-building interact inseparably. Choices regarding the desired initial capabilities determine what knowledge to acquire first and how to engineer it for use. Over time the knowledge base expands to support additional capabilities, and this expansion often strains the capacity of the initial knowledge formulation. Thus the knowledge engineer frequently reaches a point where future progress depends on improved conceptualization and related reformulation of that knowledge.

One of the basic facts of the knowledge engineering process is that it is iterative in nature; in 4GL terms, we apply a prototyping methodology. Since Brant has been involved in 4GL consulting for several years with Powerhouse, Speedware and

	HP 3000	A102/8
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Rapid we felt we could "wing it". We did rely on experts from Logicware Inc., the developers of MPROLOG, for consulting assistance during our startup - something we strongly recommend to others. Although we can all "go it on our own", the insight of an experienced knowledge engineer can be worth its weight in gold.

4.1 Task Suitability

As outlined in Section 3, the selection of the appropriate task is essential to a successful project. Without limiting what we discussed earlier, the main points to consider are:

- Focus on a problem area that does not require "common sense" to solve;
- The task should be very clearly defined, and one that is solved reasonably effectively by experts;
- Get a solid commitment from your expert, whether inhouse or not.

4.2 Building the First Prototype

From our experience it is first necessary to familiarize yourself with the problem domain by reading books, literature or articles on the area, or else through extensive dialogue with your expert.

Once you are feeling comfortable with the domain, clearly identify and characterize the important aspects of the problem. When the problem has been clearly circumscribed, use the observational method to record the expert solving one or two cases that he feels are typical of the problem area. We addressed the area of knowledge engineering in Section 2.

Now comes a crucial stage in the project: the selection of the appropriate tool. Many projects falter or fail because the tool selected is not appropriate for representing the domain concepts and control structure. In such an event change tools! Do not try to make the tool do something it is not designed to do. A second aspect of this problem can be that since the field itself is young, many tools are new to the market or immature. Try and pick a tool that has a good

track record so that you don't end up spending more time debugging someone else's product than working on your own project.

When the tool has been selected and the first example is well understood, start building the first prototype. Try to have this working within a couple of months.

Focus on a limited number of representative problems. Make sure your system accurately handles these before branching out. Avoid getting bogged down in problems that the whole AI world is still struggling with (time and space relationships, natural language, etc).

Try to keep the domain specific knowledge separate from general problem solving rules. Keep your inference engine simple and don't worry about efficiency in the first effort.

Try to develop or purchase tools that will help to assist in the rule-writing process. Many shells have such modules included and language vendors may provide you with examples of similar implementations.

Document the system as you would any other system. When testing the system consider the impact of errors in input/output and how they will impact rules or control strategies.


4.3 Extending the First Prototype

Once you have your system working and effectively solving your cross-section of representative problems, you are going to want to let a number of people start to work with it. A few design considerations can go a long way.

Be sure you have an explanation facility so that users can review the reasoning that leads to a conclusion and the facts that are in the knowledge base.

Some very simple "front-end" interface features will help your users make better use of the system. Don't expect to train everyone on the "vocabulary" of your system.

Provide a mechanism for the users to record their comments or

	HP 3000	A102/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

complaints if the system doesn't work. Quite often observations about the accuracy of the rules of the effectiveness of the system will not be recorded unless you are standing over their shoulder or such a facility exists.

Keep a test library of cases so that you can test against them as you continue to refine the system.

It is important to keep the expert "onside" during the process. Keep him sheltered from the technical problems but involved in such aspects as designing the interface.

Be careful that in all you have learned you don't start acting like the expert yourself and become challenging. Quite often, having built in a line of reasoning the knowledge engineer may become "protective".

4.4 Building the Second Generation of the System


Throw away the prototype. The purpose of the prototype was to refine the rules into machine utilizable form and gain a better understanding of the domain. This has been accomplished so don't saddle yourself with the mistakes from the first design.

Begin to consider questions of generality of problems and the ability of the system to respond effectively. Performance should become more of an issue with larger production systems.

Decide who the real users of the final system will be and make sure that the system I/O will feel natural to them. What may have been sufficient for yourself and the expert may scare others away.

4.5 Evaluating the System

Early in the process you should confirm with the expert how he would evaluate the system to determine whether it was useful. Does it match with what you are trying to have the system accomplish? Are you solving a problem that will be of value to your users, or will it become self-evident after the

	HP 3000	AI02/11
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

first few times they use the system?

As an aside, in one documented knowledge engineering exercise, it was discovered that the expertise of one company's expert could be refined down to about twenty rules. Although an exception, in that case the expert was devastated and quit.


The user interface is crucial to the acceptance of the system.

5.0 Summary

Rather than attempt to design a flowchart that shows arrows going from knowledge engineers to experts to prototypes and so on, we have tried to outline many of the general concerns that are part of the "methodology" of developing an expert system. If your firm already has standards for documentation and reporting, they will probably be applicable with little modification.


The most important fact to remember is that although expert system development is not "research lab material", it is inherently experimental at this point. Take the time to familiarize yourself with several other systems and projects before you get underway. This is one of those occasions where front-end study and reading will pay dividends overall.

January 1987

	HP 3000	AI02/12
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Footnotes

1. A Guide to Expert Systems, Donald A. Waterman, Addison-Wesley Publishing Company, 1985. Page 154
2. A Guide to Expert Systems, Donald A. Waterman, Addison-Wesley Publishing Company, 1985. Page 157
3. Building Expert Systems, Hayes-Roth, Waterman & Lenat, Addison- Wesley Publishing Company, 1983. Page 127


	HP 3000	A103/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

LOGIC PROGRAMMING AND EXPERT SYSTEMS

Shawn Brayman

Brant Computer Services Limited
6303 Airport Road, Suite 201
Mississauga, Ontario
Canada

(416) 673-9417

	HP 3000	AI/03/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

1.0 Introduction

This paper is intended to provide an executive summary or overview of Artificial Intelligence (AI) in general and Logic Programming in particular. I have attempted to provide enough detail and examples to make the paper meaningful while at the same time keeping it geared to an introductory level.

Sections 1 and 2 are a short overview of AI and a capsule history of the field. In section 3 we will take a quick look at AI programming, some myths and some observations about this new technology. Included in this discussion will be a mention of the pros and cons of LISP versus PROLOG as the two primary development languages in the AI world.

In section 4 we will take a look at the knowledge acquisition process and try and differentiate between the role of a systems analyst and a knowledge engineer. Section 5 reviews a few of the specific expert system projects Brant is involved with and the final few sections discuss the AI marketplace, and HP's positioning in that market.

2.0 Overview of Artificial Intelligence

Artificial Intelligence is not a new field - it has been around since the 1950's when people like Dr. Marvin Minsky helped found the first Artificial Intelligence Laboratory at MIT. Since that time thousands of researchers in dozens of universities have added to the research effort. Although the field is not new, what is new is the perception of commercial readiness of certain aspects of the technology, namely expert systems.

For the first decade, AI research appears to have been down what proved to be a blind alley. This effort attempted to create a hardware/software "thinking machine". It was felt that once we discovered how a person thinks, we could put this "thinking algorithm" into a computer. We could then provide it with information and it could "think" out the answer. The search was on for a general problem solver, a thinking machine. The search was unsuccessful.

By the mid 1960's, recognizing that the attempt to create a

general problem solver was not going to be successful in the short term, researchers tried a new tack - to try and create a program that could emulate a human expert in a limited domain of knowledge. The first major expert systems were born in the late 1960's.


DENDRAL was an expert system designed to help determine the structure of chemical compounds based upon analysis of the components of the molecule. MACSYMA was a second expert system started shortly afterwards that was designed to solve symbolic mathematical problems, much as we did in algebra back in high school. In both cases, however, the difficulty of the problems is substantially greater than a high school level. Both of these expert systems are in routine use today.

When people discuss AI, confusion tends to arise as a result of the various aspects of the field - in the same way that the question "What is computer programming?" could be met with answers ranging from machine-level programs to applications programs, with a world of possibilities in between. In the field of AI answers are further complicated by the fact that the linguists, the psychologists, the philosophers and the computer scientists working in AI all have different perspectives. Some of the application areas under study are:

- expert systems
- natural language understanding
- automatic programming
- learning systems
- perception and vision recognition
- robotics

When we discuss whether or not any of the existing AI systems are truly "intelligent", all of the experts, the AI gurus, disagree. Some feel that we can already emulate intuition and human intelligence in some areas. Others, like Minsky, feel we are probably fifty years from machine intelligence and must first teach machines things like common sense and a sense of humour.

I will not take sides in the argument, nor do I feel it is that important. Most of us have enough trouble determining if there is intelligence in many people, let alone machines.

	HP 3000	A103/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

What is important is that whether intelligent or not, we are developing a new style of software that in some ways emulates human "thinking". There appears to be little doubt that the impact of this new type of software will be substantial.

Now let's take a look at some specifics about AI programming.

3.0 AI Programming

Although it is obvious that there must be differences in AI programming (else why the hype?), there are obviously as many myths about AI. In this section of the paper, we will initially discuss several basic issues concerning AI. From there we will go on to discuss some specifics of AI programming and how it is advantageous.

First let me emphasize that AI programming is, most importantly, programming. You have a language, usually either LISP or PROLOG, and you develop programs. The programs we will be discussing are intended to solve certain types of problems - problems that are usually solved by someone in your organization with specific types of expertise; an expert in his field.

Second, although AI is often called fifth generation and PROLOG and LISP specifically called fifth-generation languages, this is a bit of a misnomer. LISP was invented in the 1950's, around the same time as FORTRAN. It processes symbols rather than data, but other than this qualitative difference, they are both of the same basic "generation" of programming tools. PROLOG was developed in the 1970's and may be more equivalent to a 4GL like PowerHouse or SPEEDWARE. Although fourth-generation languages were or are supposed to replace third-generation languages, symbolic processing is not designed to replace 4GL's.

In fact, one of Brant's current projects is the development of a Personal Financial Planning System in conjunction with a leading financial planner in Toronto. This system is actually composed of two parts: a conventional 4GL financial model developed using SPEEDWARE and an Expert System Financial Advisor that incorporates the financial planning expert's knowledge of how best to manipulate the model. We

are in effect applying the strong suits of a 4GL language to those aspects of the problem which are algorithmic in nature, and using Brant's MPROLOG product for the expertise-based aspects of the system.


Third, let me point out that any program that can be written in MPROLOG or a 5GL can be written in a traditional language like COBOL. In fact, MPROLOG is written in the "C" language on the HP3000. With this in mind it stands to reason that anything written in MPROLOG on the HP3000 could be written in "C".

Most AI programs are a series of rules and facts expressing the relationship between any number of things in a problem domain. The rules express these relationships in simple forms:

- Rule 1: The computer won't work if
there is no power.
- Rule 2: The computer won't work if
the power supply is cut off.
- Rule 3: There is no power if
the on/off switch is turned off.
- Rule 4: There is no power if
the cord is not plugged in.
- Rule 5: There is no power if
the building's power is out.

In creating a diagnostic program to determine why our computer won't work, we put in a series of rules that outline relationships. MPROLOG will then sort and order these relationships, much as you might structure them in an ordinary program.

Let's take a look at some specific examples.

	HP 3000	A103/6
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

3.1 Dealing with Complex Relationships

In an effort to outline a few of the areas where AI techniques and tools are advantageous, we will look at a few examples. First let us consider a situation where a relationship exists between several conditions. The example we will use is that of a program to determine if an individual is eligible to receive a personal loan. Let us assume that the factors that are considered in a loan application are the amount of the loan, monthly payments, income of the individual, security of the loan and stability of the individual.

In MPROLOG we may have:

```

eligible (name, amount, payment, income, security,
          stability) if
capable_payments (amount, payment, income) and
loan_size_OK (amount, security) and
stability_OK (stability).

```

Each of the relationships to determine stability and the capability to make payments may in turn be a function of other factors or rules. We may have a query on our program of the sort:

```
?eligible(Shawn, 5000, 300, 2000, 0, none).
```

This query is asking if Shawn is eligible for a loan of \$5000.00 with payments of \$300.00 per month if he makes \$2000.00 per month, has no security for the loan and is an unstable sort.

At this stage, we have seen very little that couldn't be easily accomplished with any other languages. Now let's try a few other inquiries.

```
?eligible(WHO, 20000, __, __, __, __).
```

This inquiry would give us a response of anyone who is eligible for a \$20000.00 loan. Another example query might be:

```
?eligible(Shawn, HOWMUCH, 300, 2000, 0, none).
```

This inquiry would tell us how much of a loan Shawn would be eligible for, given that he only wanted to pay \$300.00 per month and the other personal information is the same.

The important thing to recognize is that once the relationship has been defined, each of the different types of inquiries all use the same code - instead of having to write a routine to calculate the size of loans people are eligible for and a routine to list groups of people eligible for certain types of loans, or just to confirm if someone is eligible or not.

In effect, every argument in a relationship may or may not be defined. This means that achieving the same degree of flexibility in a traditional program would require 2^N routines. Obviously not every real world situation requires this degree of flexibility, but given a complex situation, the advantages become clear. In our example above, 64 separate routines or procedures would be required to accomplish the same effect as our one relationship in MPROLOG. In other words, you must think out all 64 routines.

3.2 Dealing with Complex Inter-Relationships

A second aspect of the strength of AI languages is the way in which we can handle a series of complex inter-relationships. Let's take a look at another example.

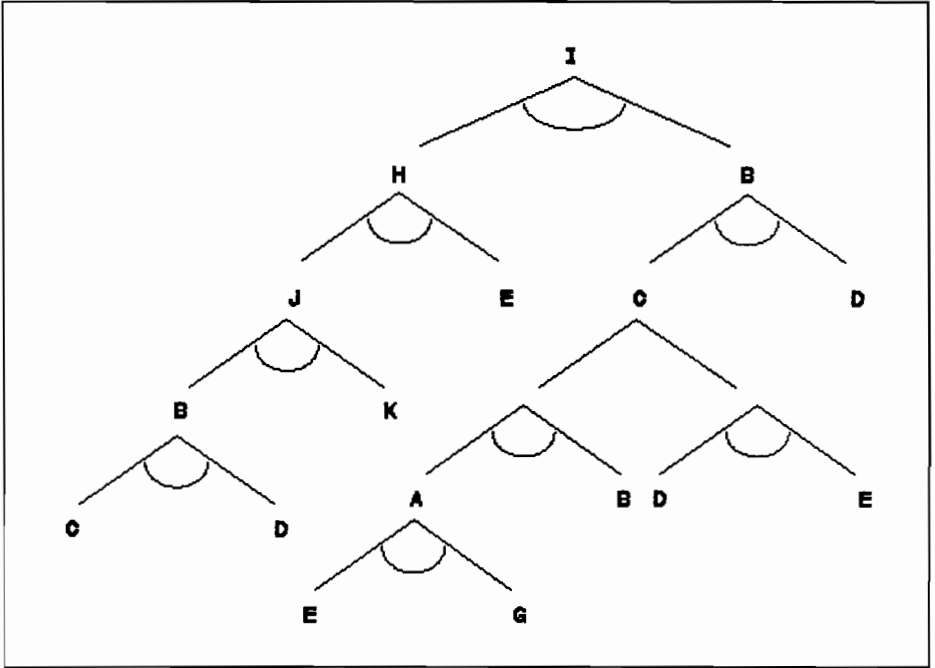
Consider the following relationships:

C is true if A is true and B is true or
 C is true if D is true and E is true
 A is true if E is true and G is true
 B is true if C is true and D is true
 J is true if B is true and K is true
 H is true if J is true and E is true
 I is true if H is true and B is true

Given that D, E and K are true, is I true?



In a traditional problem-solving style, we would work out the relationships and develop some form of "tree-like" structure such as:



We would then encode this "solution structure" into our program in a series of structured IF/THEN's, LOOPS or whatever appeared necessary to best address our problem.

Now consider the problems we may encounter if we increase the number of relationships (we only have seven listed), if a relationship needs to be removed (say the relationship of H

to J and E) or we need to be able to solve to find any of the ten conditions at any time. As we can perceive, with any of these scenarios, the complexity of our program would increase dramatically as would problems of maintenance.

To write the same program in MPROLOG, we would have:

```

true (c) if true (a) and true (b)
true (c) if true (d) and true (e)
true (a) if true (e) and true (g)
true (b) if true (c) and true (d)
true (j) if true (b) and true (k)
true (h) if true (j) and true (e)
true (i) if true (h) and true (b)

true (d)
true (e)
true (k)

```

Our inquiry would take the form:

```

?true (i)
YES
?true (g)
NO


```

To add new relationships or remove them, we simply add or delete that line of code. The order of the relationships does not matter, nor which condition we are searching for. In effect, we define the problem as opposed to the solution. We let MPROLOG automatically create our "tree-structure" and work out the inter-relationships. We do not need to worry as we add and delete new relationships.

3.3 Knowledge Trees and Heuristic Search

We seem to have described a system that will automatically generate a tree structure according to the relationships and allow us to traverse it, but what does this mean? There are other forms of decision support tools that use binary trees and there is no "magic" to them.

The qualitative difference with our Prolog system is not in the underlying tree but in how we traverse it. Picture a

	HP 3000	A103/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

tree that becomes so large, with so many possibilities and permutations that passing through takes substantial time and resources. It is at this point that we introduce the concept of search techniques with backward chaining, forward chaining, breadth first search, depth first search, applying weights or probabilities to events and any number of other methods of searching a set of facts.


We also apply "rules" or heuristics - much as a human would - to cut down the search space, rather than try every possibility. If we are trying to leave a room, we do not try the floor, the ceiling and the walls; we locate a door or window, locate a latch or knob, etc. Humans have rules that permit shortcuts rather than undergoing an exhaustive search for possibilities.

3.4 How do I know what is happening?

One of the obvious things about an AI program is that it is designed to automate an area of expertise that is not "obvious" in content, else we wouldn't call the person who does that job an expert. With this in mind and given the examples we outlined above where a small piece of code can be used 64 different ways, how does a programmer know what his or her program will do? The answer is, he doesn't.

AI programs are designed to address real-world problems with non-obvious answers, so we must expect that once an AI program reaches a certain level of complexity, your programmers will not know what the system will do. Experience with some of our own projects has demonstrated that with as few as half a dozen relationships and a few variations of each, you will be surprised at some of the ways that your system will react.

This raises two key issues concerning AI programming. First, if you don't know, you ask your program "WHY?" Unlike traditional programs where you know it is doing a process or procedure because you told it exactly what to do, in AI systems we don't know. The solution is that your expert system can be coded in a manner so as to allow you to ask WHY at any stage, and the system will outline the rules that it used to reach its conclusion.

	HP 3000	AI03/11
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

In our previous example using the computer for instance, you may tell the system that your HP3000 won't work. It may respond with: "Check to see if the on/off switch is turned off". If you ask it "WHY?" it would respond:

Because - Rule 1: The computer won't work if
there is no power, and

Rule 3: There is no power if
the on/off switch is turned off.

A second benefit of this ability is that it exemplifies the value of AI as a training and educational tool. The fact is that when your company encodes the expertise of someone in an AI system, this expert system can be used both in production and in training less experienced people, if you wish to design your system to do so.

3.5 LISP versus MPROLOG

In Brant we still have "discussions" among many of our people as to the advantages and disadvantages of COBOL versus fourth-generation languages. A similar argument exists in the AI world between LISP and PROLOG users.

People wishing to develop applications in AI have three choices: one of the two base languages, (LISP or PROLOG), or what are called expert system shells. Expert system shells are partial expert systems already in place, where you drop in the specific rules you want and have a working expert system in less time and at less cost for man hours than with the base language.

According to the March issue of a newsletter entitled Artificial Intelligence Markets (AIM) published in the United States, "several high end language vendors reported the return of customers who went off to buy expert system development tools to AI language products because they found the tools inadequate for their needs. Since sixty to seventy percent of high end AI language customers are Fortune 1000 firms or the federal government (as opposed to universities and AI companies), this is not a trend to sneeze at."

Accordingly, we will leave this discussion focussed on the two AI languages and leave discussions of the tools to others. In a tutorial on PROLOG presented at the International Joint Conference on Artificial Intelligence in August 1985, Dr. William Kornfeld pointed out several "cultural" and practical differences between LISP and PROLOG, namely that LISP is all-inclusive, has complex semantics and hence a large manual that can be confusing. PROLOG strives for simplicity, doing a number of things very well, such as symbolic structure manipulation and processing facts, and consists of a few well-chosen constructs. In a practical sense, LISP has a few features PROLOG does not, like destructive assignment and structure manipulation, iteration with DO's or LOOP's and global variables. PROLOG has logical variables and efficient procedure calls that make the programs clearer and easier to understand, at the expense of some generality.


Again, according to the AIM newsletter, "We think LISP will evolve into a systems manager language - system management functions will be in LISP, and they will be layered between the operating system and the applications. PROLOG and other languages (including conventional languages) will serve as secondary, application languages which will be called from LISP."

By no means is this intended to resolve any debate, but rather it is intended to outline some of the issues and where things appear to be heading. Brant at this time is only involved in application work with MPROLOG.

4.0 The Knowledge Acquisition Process

An important aspect, actually a critical aspect, of developing an expert system is the process of knowledge engineering. This is, in effect, the new name that we give to systems analysts to confuse things. A knowledge engineer plays the same role as a systems analyst, with a few important differences.

A knowledge engineer is an individual who attempts to discover the "rules" that a human expert makes to reach conclusions in problem solving. A traditional systems

	HP 3000	AI03/13
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

analyst may ask a user specific questions about an application and hope to get a reasonable answer. Knowledge engineers are warned on the other hand not to expect a reasonable answer and in fact not to take what the expert says at face value.

Research has uncovered a phenomenon known as the "Expert Paradox": the better your experts, the worse they will be at describing how they make decisions. It sounds like a joke, but unfortunately it is true.


It appears that our "experts" use what can be called "compiled expertise", where they may take dozens of specific rules or conditions and relationships which they look for in a specific instance, and compile them all down to one specific rule. It is the knowledge engineer's task to "decompile" this expertise to a form that is usable in our automated systems. The problem is that when asked, your expert will not recognize that some of his "rules" are actually compilations and so will provide the knowledge engineers with inaccurate or misleading information.

The selection of a capable knowledge engineer is a crucial part of the expert system building process. There are many good books describing knowledge engineering and expert systems, a few of which are listed in the bibliography of this paper.

5.0 Expert Systems

Having belaboured what AI is all about, some aspects of how it operates and some discussion of the tools we can use, it's about time we got down to why and where you use it. Why develop an expert system and how is it different?

Rather than trying to provide infallible rules of thumb, I will review briefly some of the projects currently underway at Brant and a few we have been discussing with clients and other prospects. I will attempt to outline why an expert system was the best solution and to what point each project has evolved.

	HP 3000	A103/14
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

5.1 Personal Financial Planning System

In December of 1985, Brant was approached by one of the top financial planners in the Toronto area in conjunction with a Canadian insurance company to create a personal financial planning system. The system was intended to allow the insurance company to provide a standard high level of personal financial planning to the customers as part of their service. The software would ensure certain levels of depth and professionalism in the analysis.

After meetings with the financial planner, it was discovered that there were two aspects to the system. First, the planner had developed a financial model that generated cash flow projections for the individual up to and after retirement. The model was comprehensive and "algorithmic" in nature, so could easily be developed using a conventional language. We selected SPEEDWARE, a 4GL that would allow us to deliver the solution on either HP3000's or Vectras with no problem.

The second aspect of the system involved the manner in which the financial planner manipulated the variables in the model. This would include factors like retirement age, return on investment, risk tolerance, portfolio arrangement and much more. We discovered that the planner used such considerations as "Who appeared to be the decision maker on investments?", "Who is the primary breadwinner?", "Who appears more knowledgeable?" and a myriad of other factors in determining a family's overall risk tolerance on investments. Decisions as to whether the client would react more favourably to retiring later at a desired income level instead of retiring when he wanted to with less money, were all decisions made from a large number of what we called "soft facts".

This part of the system has been written in the MPROLOG language, a product developed by Logicware Inc. in Toronto and distributed on the HP3000 by Brant. Like SPEEDWARE, this language is available on the HP3000 and the Vectra, allowing us to deliver a total system on both micros and 3000's. This Planning Advisor is the "expert system" that has been developed by Brant to complete our Financial Planning System.

5.2 Expert System Manager

Brant currently has thirteen HP3000 computers in our various offices with operators and system managers in the necessary offices. In a number of instances we have had to bring a senior systems manager to a specific project, leaving more junior support people in charge of a facility.

When certain problems arose (as they inevitably do), the operators in an office would call their systems manager who was somewhere across the country, and resolve the problem through a long-distance dialogue. After witnessing the process of question and answer strategies to resolve problems remotely, we concluded that this was an ideal example of an "expert system" with a live expert. As an internal project, we undertook the development of a limited expert system manager for the HP3000.

5.3 Irrigation Event Scheduler

In conjunction with a firm in Fresno, California that is acting as a beta site for MPROLOG, Brant has developed a prototype "Irrigation Event Scheduler". Professional Agricultural Management (ProAg) provides specialized management consulting to farmers in the San Joaquin Valley on such things as irrigation scheduling, fertilizer application, pesticide application, crop rotation and more.

In some instances, decisions are reached through an algorithmic application of soil measurements, climatic data and historical tabled information, but at the same time, judgement calls on crop vigour, irrigation method, management constraints require that schedules be "revised".

Our prototype system combines the algorithmic considerations and "soft factors" to arrive at a preferred irrigation schedule for a specific field.

5.4 Other Expert System Projects

At this time (January, 1987), Brant is involved in a number of major conversion projects. One aspect of the application

of AI that we are currently studying is its use in writing software program conversion systems.

When we convert one version of COBOL on a foreign architecture to the HP3000, we are in effect going from one language to another, or more accurately between two versions of the same language. In so doing, we are "processing symbols" or in this case, our program syntax.


Although we have not yet implemented any conversion tools, discussions with Logicware, Inc. of Toronto, Canada have indicated the achievability of some distinct advantages using this approach, based on their own development of converters between different versions of PROLOG.

Another area where a number of feasibility expert system prototypes have been developed is in the Risk Assessment area. Specifically, some of the major banks in Canada have been looking at commercial loan risk assessment systems that would allow some degree of consistency in the processing of loan applications. The prototype captures many of the factors that the banks' top loans evaluators use in assessing an applicant, including management competence, business environment and marketing ability, as well as standard items like the balance sheet. For large commercial loans in excess of \$5 million, this process also ensures that the branch will address all issues prior to referring the application to head office.

There are hundreds of existing applications already in use and, as we are all aware, substantial amounts of effort going on behind the scenes. With all of the information that has been published, I will not prolong the discussion on this point.

6.0 The AI Marketplace

Of interest to all of us must be the rate at which this technology is impacting the marketplace and the "real world". The two major indicators that are available from various market research studies on AI are the dollar volumes of products on an annual basis, and the number of "units" that are delivered, given that the unit price of AI products will

	HP 3000	AI03/17
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

be dramatically declining, much as is true for micro-electronics in general.

In many areas, we can expect that "intelligence" will be built into other products and will not be covered by the studies outlined herein. As an example, if intelligence is built into an electrocardiogram, it would not be included in the AI marketplace, but rather is a byproduct. The magnitude of this secondary market is difficult to discern at this time.

According to a report by Frost & Sullivan entitled "Artificial Intelligence Products", printed in the January, 1985 issue of Computing Canada, the hardware, software and services marketplace for Artificial Intelligence nearly doubled from \$181 million in 1984 to \$342 million in 1985. It is expected that it will double again by 1987 to \$665 million and will reach a total of \$1.6 billion by the end of the decade.

Of this \$1.6 billion, 50% will be software products, 30% hardware and the balance services. The software component will be comprised of 20% natural language software that will be popular on mainframes and personal computers, due to the higher proportion of novice users. Expert systems are expected to comprise 35% of the total software sales, 31% will be AI languages and the balance will be AI applications.

More important than the growth in dollars of the marketplace is the fact that as hardware prices continue to drop, more units can be expected to be delivered, leading to a projection that "the AI market will multiply by a factor of 40 between 1984 and 1989".

According to a projection on AI languages specifically by Artificial Intelligence Markets, between 1985 and 1990 we can anticipate a growth in sales from \$32 million currently to \$226 million in 1989. This agrees relatively closely with the Frost & Sullivan study (see Table 1). During the same period, the number of shipments of AI language packages is expected to grow from 5,000 in 1985 to 50,000 per year by 1990.

Frost & Sullivan

	1985	1990	
Expert Systems		280	
Natural Languages		160	
AI Languages		248	
Applications		112	
Total Software	137	800	484%
Hardware	85	480	464%
Services	120	320	167%
Total AI	342	1600	368%

Table 1: Growth in mil's of the AI Marketplace, 1985 to 1990

7.0 HP and the AI Way

In this paper for the Hewlett-Packard users group, we would be remiss if we did not do a quick review of what is happening in the HP world in particular.

To begin with, let me say that HP probably made the most impressive showing in the 1985 International Joint Conference on Artificial Intelligence held in Los Angeles, and had one of the larger vendor presentations at AAAI in Philadelphia in August of 1986. From a totally non-participatory role in previous years, HP has become one of the top two or three vendors with their new HP9000 Series 300 AI development environment.

This hardware and software environment, running under HP-UX, amazed many attendees at the conference with its quality, thoroughness and presentation. HP released the new 9000 system with a full LISP development environment complete with expert system shells, tools and more. By any standards the offering was impressive and HP's commitment to the AI field notable. A PROLOG language was implemented in conjunction with a Third Party from Switzerland in 1986, which "sits on top" of HP's LISP environment.




On the HP3000 there has been a noticeable lack of attention to the AI field, primarily because most people saw the potential for AI in the engineering and technical environment as opposed to the commercial world. For the past few years, the only AI tool available was a version of LISP from Robelle in British Columbia that provided a learning tool, but which was not robust enough for a proper production environment. I strongly recommend that anyone interested contact the people at Robelle.

Because of the lack of tools and rumblings of interest from commercial users, Brant entered into an agreement with Logicware Inc. to port their MPROLOG development environment to the HP3000. At the time of this paper, the port is complete and should be in beta by February of this year. Because MPROLOG runs on the Vectra, application code will be transportable between IBM-PCs and the HP3000. Brant has been able to commence work on our expert systems prior to having the interpreter on the HP3000. MPROLOG will be written in "C" on the HP3000 using the C.C.S. "C" Compiler.

Of specific interest to Brant has been the role of AI on the Spectrum, and although definitive statements have not been forthcoming from HP, the rumour mill is churning up good things. In the July issue of High Technology magazine, Ira Goldstein, director of the Distributed Computing Center at HP's labs in Palo Alto, was quoted as saying: "You want machines that can do symbolic computing, but not at the expense of conventional computing...and that's where HP's forthcoming Spectrum line is a step in the right direction. The new computers will have a large address space, a key requirement for knowledge-intensive AI programs as well as a large number of storage registers, which can hold functions commonly used by the LISP language. Perhaps most importantly, the Spectrum computers are designed to support coprocessors of different types. Many observers believe that the hybrid machine of the future will have general purpose and LISP microprocessors working together to run the numeric and symbolic portions of mixed applications!"

Hardly a lot to go on, but based on the smiles of many of the people in the HP labs we feel optimistic that HP's new precision architecture will lead the way for those interested in this new technology.


	HP 3000	A103/20
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

8.0 Conclusion

It's hard to apply a moral to the story, but I do hope this paper has been successful in providing those interested in AI with a little more insight into some of the issues in this area.

This paper has been geared to people new to the field and thus does not address some major areas. For others with more specific interests or inquires, we at Brant would be more than happy to talk with you.

January, 1987


	HP 3000	A104/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

An Expert Financial Planning System

Don MacKenzie, Ross Hopmans, Shawn Brayman

Brant Computer Services Limited
6303 Airport Road Suite 201
Mississauga, Ontario

(416) 673-9417

	HP 3000	A104/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

1.0 Problem Description and Overview

This paper discusses a financial planning package developed by Brant Computer Services that combines 4th and 5th Generation Languages in a financial service oriented expert system.

The mandate from our client, an experienced financial planner, was to design a micro-computer based system that provides two distinct yet integrated functions:

- 1) A package that stores the client's personal and financial data. Once the data is in the system, it must be formatted into the necessary financial statements, schedules and accompanying calculations intrinsic to the financial planning model.


- 2) The capability to provide recommendations based on quantitative and qualitative information about the client being evaluated. The knowledge required to make these recommendations originates from the financial planner and this knowledge is represented to the expert system.

The interesting implications of this project stem from the practicality and feasibility of expert systems in the financial services sector and the degree to which important business and financial decisions can be formalized into a computer based system. Related to this issue is the question as to whether the system can be economical, user friendly, and portable.

Secondly, the attempt to deliver a system for production use which is a hybrid of an artificial intelligence program and a conventional fourth-generation language program has proven to be a challenging and exciting process.

2.0 Design Approach and Analysis

Initially we defined the two distinct functions of the system. As mentioned in the introduction, one component of the package involves data capture, reporting functions, and a modelling capability. These tasks are typically

	HP 3000	A104/3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

straightforward and formalized and they must be carried out by the financial planner in order to make strategic decisions based on the financial and attitudinal attributes of the client. To put it more simply, this is the 'front end' of the financial planning process: the accumulation and integration of the client's financial information.

The second (and more elaborate) function of the system is the 'expert' capability that involves the integration of both the quantitative and the qualitative factors in the financial planning process (more about this later). This functionality implies a less tangible process than the one that occurs in the front end section. However, a similarly formalized procedure is required on the expert side although the decision criteria and considerations for the expert process are broader and involve factors that are very subjective.

Our aim here is to let the financial planner provide the rules and reference points to the expert system so that the expert system can supply valid strategies for the client to follow. Herein lies the obvious challenge in this type of project: the formalization of the complex and seemingly non-procedural processes and the duplication of these processes on a computer based system.

The identification of the 'front end' and the 'expert end' of the system gave us a starting point in the design of the system. By determining the data requirements for the reporting functions and the higher level processes, we were able to design the 'front end' section of the system.

2.1 System Specifications

The system front end was designed to handle data entry, reporting and modelling functions. The primary functional requirements of this system were defined from a number of sources. Primarily, our client provided us with samples of the input documents for his clients as well as sample plan presentations. The client was generating the bulk of his reports and schedules through a LOTUS 123 spreadsheet.

On top of this, we received a number of plan presentations from other financial planning services. The exposure to the

various planning approaches and presentations enabled us to extract the common requirements of the different planning philosophies.

We attended a financial planning conference and paid particular attention to the different planning software packages that were being promoted at the conference. Our emphasis here was on the capabilities and user friendliness of these packages.

Critical to our analysis was the extensive amount of time spent with our client on the financial planning process in general. We did have access to a broad range of financial planning material so that we were able to get an idea of the amount as well as the nature of the information involved in the financial planning process itself.

The formalization of the planning process was achieved by sitting down with our client, extracting his knowledge, and formulating a set of rules to guide the expert system. This 'knowledge engineering' involves refining the expert's knowledge into a series of relationships and conditions that can be utilized by the expert system. It is this information that guides the decision process.

3.0 The Financial Planning Process

Before describing our particular model, I will briefly discuss the financial planning process in general. Essentially, this process involves the assimilation of all the aspects of an individual or family's financial and personal status. With this information, the planner devises a methodical strategy whereby the client can achieve his/her specified level of financial independence. This sounds relatively straightforward, although it will become apparent later in this paper that the process is complex and involves many variables.

Currently, the financial planning industry is unregulated and fragmented. Financial planning, unlike other financial services such as accounting, does not enjoy a uniform, standard methodology. Depending upon the financial planner involved, the quality, method, and philosophy behind the



planning does vary. On top of this, financial planning services may be industry driven (the service may be an arm of a financial institution such as an insurance company or investment dealer) or product driven (the planner favours a particular investment vehicle).

The purpose of this paper is not to perform a critical analysis of the industry but rather to highlight the fact that there is a need to standardize and define some of the universal requirements of financial planning. One objective in designing a package such as this is to provide a front end to the expert side of the system that is relatively devoid of one particular approach. In other words, the facts and information that are supplied to the expert system should be unencumbered by a particular planning philosophy and the expert system should not be constrained by a limited amount of information about the client.

3.1 The ROGI Model

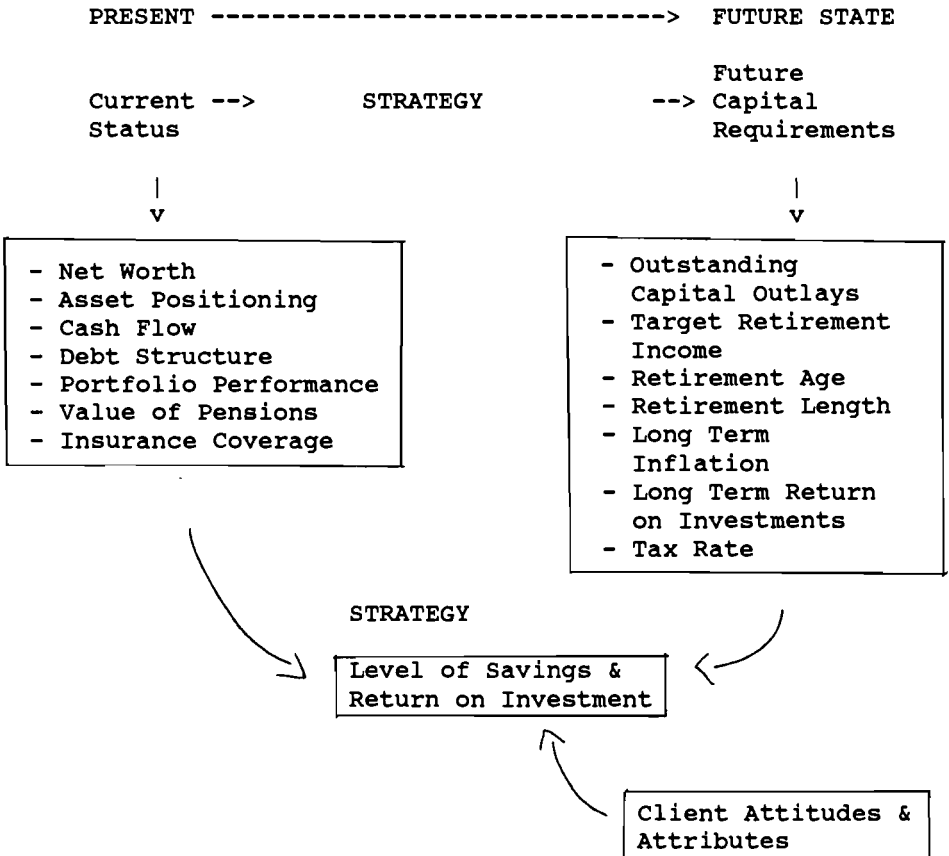
The philosophy behind the Rate Of Growth on Investments (ROGI) model is simple and easy to understand. The aims of this model are as follows:

- 1) determine client's current financial status.
- 2) determine future capital requirements and objectives.
- 3) using a strategy of savings and return on investment, enable the client to achieve the targeted financial goals within a time frame that is both suitable and realistic for the client.

The key philosophy behind the ROGI model is that the achievement of the client's financial objectives take into consideration several of the aspects of the client's financial and personal status. This means that the optimal strategy involves accounting for asset management, cash flow, investment planning, estate conservation and distribution, and the client's attitudes toward financial independence and lifestyle.

3.2 ROGI Strategies

In the ROGI model, the client's future capital requirements are laid out in such a way as to enable the planner and the client to relate the client's specified financial target to the client's current financial status. If the client is already on a path that will allow him or her to achieve the desired financial goals, it is unlikely that the individual is in need of a financial planning service. If, on the other hand, the client must alter current savings, portfolio performance or retirement expectations, then a strategy must be devised to enable the client to achieve the target capital pool at retirement.



Looking at the diagram on the previous page, we can get an idea of the types of considerations that come into play in the planning process. The front end of the system will provide the planner with the storage of the data, reports required from the hard data and the impact of the predictive data on future capital requirements.

The ability to assess the impact of the predictive data is what I have referred to as the modelling function of the front end of the system. By modelling, I mean that by manipulating certain variables (they are quantitative) the planner can see the resulting sensitivities on the capital requirements of the client's financial goals. This is important since there is not necessarily one optimal savings and investment rate, and the planner must be able to determine the impact of alternative scenarios on the client's future capital requirements.

With the client's current status and projected requirements in hand, the planner and the client can get an initial impression as to the viability of the client's goals. The planner can begin to see how the client's current cash flow and portfolio fit in with the level of savings and investment that will lead to the desired retirement income.

OPTIMIZE SAVINGS

1. Earn more
2. Spend Less
 - budget
 - restructure debt
 - alter living standards
3. Taxes
 - shelters
 - deferral strategies

OPTIMIZE RETURN ON INVESTMENTS

1. Increase Risk
2. Investment Education
3. Professional Management

Given that a client has specified a retirement income and this is translated into a target capital pool, an optimal savings and investment return are required. How are these levels determined? Obviously, the client's projected cash flows and portfolio performance are determinants. On top of this, the client has certain attitudes, opinions and biases



that prevent particular combinations of savings and returns on investments from being viable planning strategies for the planner's client.

What are these constraints? These are considerations in the planning strategy that are not purely economic or quantitative. For example, the planner may recommend a target savings level for the client to follow that is within the realm of the clients cash flow. The client, however, may find this savings level unacceptable in that it reduces his or her immediate standard of living to a level that does not justify the purpose of the savings plan. In the same manner, a suggested portfolio performance may be rejected by the client on the grounds that it implies a risk tolerance above that of the client. Or the planner may recommend a restructuring of assets to increase liquidity that the client cannot accept because the asset in question has a 'sentimental value' to the client.

4.0 The 4GL Component

The previous discussion has given an indication of the considerations involved in the financial planning process. Despite the repeated warnings about the intangibility of much of the data, there is obviously a need for a variety of financial statements and reports based on quantitative information. An objective of the 4GL component of this package is to integrate information and give a profile of the client's financial status.

On the asset management side, the key reports to be produced are statements of net worth, asset positioning, and the portfolio performance of the the client. Specifically, the net worth information relates the debt/equity aspect of the client's holdings. Asset positioning gives a breakdown of the types of assets held by the client (personal, invested capital, tax shelter etc.), as well as the liquidity of the assets. The portfolio profile is useful in determining if the individual is adequately diversified and has a portfolio that is in line with his or her comfort level with regards to safety of capital. The portfolio profile can also indicate how each component of the portfolio is performing relative to the portfolio as a whole.


Cash flow reports indicate to the planner the ability of the client to reduce current debt and to increase the client's net investable capital. When determining an optimal savings level for the client, the planner must have a good feel for the client's discretionary cash flow from the present time to retirement or financial independence.

Referring to the diagram on page 4, the asset and cash flow reports profile the current status of the client. The future capital requirements of the client are calculations that are based on a combination of the client's needs in the future as well as certain predictions about long range economic conditions such as the rate of inflation, the client's nominal return on investments, tax rate on retirement income etc. Other factors to be determined include years to retirement and the length of retirement.

These financial and future requirement reports are the basis of the planning process. The client's attitudes and characteristics may have a large impact on the particular strategy devised for the client, yet these qualitative characteristics are meaningless if the economic circumstances of the client preclude a realistic chance at attaining the client's objectives. In other words, before the planner can be concerned with all of the considerations of the planning process, he or she must have the basic financial profile of the client in order to determine a realistic financial objective for the client. This relates back to the statement that while the 4GL is providing the 'less glamorous' component of this package, the overall efficacy of the expert system will be proportional to the quality of the information it is accessing, as well as the rules that make it up.

4.1 Modelling

The planner and the expert system require the capability to determine the impact of various strategies and manipulations on the client's future cash flows, capital requirements, and portfolio performance. Given that a particular savings level and return on investment are not feasible for the client, the planner will investigate the viability of various savings levels, returns on investment, retirement parameters, investment vehicles etc... The impact of a proposed scenario

	HP 3000	A104/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

or strategy must be immediately reflected in the system. Of particular importance in the ROGI model is the ability to see how various assumptions and scenarios impact on the target capital pool required by the client at retirement.

For example, if a particular financial objective is not realistic given the client's current desires and financial profile, the planner may want to determine whether a 5 year deferment of retirement will enable the client to amass the desired capital pool at retirement. Alternatively, the planner may test the impact of an increased investment return on the client's ability to achieve the required level of invested capital. Below are some examples of the modelling capabilities that the 4GL should provide:


- 1) The effect of a different tax rate on retirement cash flow
- 2) The impact of the purchase of a real estate investment on capital requirements at retirement
- 3) The impact of a deferral of retirement on the required level of savings and return on investment to achieve the targeted financial goal
- 4) The impact of a restructuring of the client's portfolio on the overall portfolio performance

4.2 Pros and Cons of the 4GL

The choice of a 4GL for the front end functions is rooted in the requirements of the front end:

- 1) strong screen handling/menu capabilities
- 2) data base access
- 3) user friendly
- 4) good reporting capability
- 5) linkup to 5GL
- 6) prototyping function
- 7) portability between HP3000 and IBM compatible
- 8) modular design capability

The prototyping capability of the particular 4GL that we used

	HP 3000	A104/11
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	


(Speedware) is an important feature in designing a package such as this one. Our client had a manual process of collecting the client's financial data and we had to streamline the data entry process to be easy to understand yet at the same time be able to capture all of the information required by a comprehensive set of financial documents. We were able to streamline the data capture process by entering the test data on the system and producing the various financial statements. In this manner we were able to play with the data entry screens and let the design stage evolve to the point where the system was capturing the necessary data while keeping the data entry process relatively clear and simple to use.

A very strong feature of the 4GL we used was a module called DESIGNER, which allows the programmer to create an application using an online menu, screen, and report writer. Changes in the input/inquiry screens of the system are reflected by a corresponding data base modification generated by the DESIGNER. Restructuring the data base as the development phase is ongoing is simply a matter of defining a new data definition at the screen or user level. The data base and code for the menu, screen and report handling is generated by DESIGNER. The approach in this type of system design is to define the system requirements from the end user viewpoint without having to be too concerned with various file structures or coding strategies.

The modularity of the system is another important consideration because if the system is designed to handle a complex scenario, it is convenient to be able to take out the more detailed functions if they are not required for that particular planning situation.

If the client only requires an asset and cash flow profile, the data entry process will entail entering asset and cash information without any prompting for future expectations or requiring specific investment information regarding tax planning, risk attitudes, desired retirement cash flows etc.

The ROGI model has one input screen for the client's personal information, one screen to handle family data, one screen for all asset types and liabilities (personal, investment, tax shelter), one screen for cash flow, one screen for future

	HP 3000	A104/12
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

capital requirements, one screen for supplementary income at retirement (pensions, annuities etc.), one screen for capital requirements at death, and one screen for insurance profiles.

The portability of the 4GL application between the mini and the micro computer enables development to occur on the HP3000 with a port down to an IBM compatible. All that is required from the 3000 environment is the download of the data base schema file and the 4GL code - and, of course, MicroSpeedware on the PC. Once on the micro, the data base is generated via a data base generating utility and the 4GL code is interpreted by the SPEEDWARE interpreter. The requirements of the micro are a minimum 512 meg storage and a hard disk to handle the 4GL interpreters and utilities.

There is a certain amount of functionality that cannot be duplicated on the IBM compatible micro. Unlike the 3000 environment, external language subroutines cannot be called. There are also certain data types that do not perform well on the micro such as data items defined as floating point integers (ie: J1, J2 fields).

The most significant shortcoming of the 4GL is the fact that it does not handle exponentiation in it's calculation routines. This posed an inconvenience in the development phase as many of the calculations performed were either present value or annuity functions. To handle this, it was necessary to write looping routines. To complicate matters further there is a calculation involving determining the Nth root of a number. While the calculations could be performed in the report language of the 4GL, the lack of the exponentiation function added overhead and processing time to the reports that require present value or compounding rate calculations.

Regardless of the difficulties, the ability to design and develop a system on the HP3000 and run it without modification on the micro with everything including an Image compatible database, is a tremendous time saver.

5.0 The Expert Component

With the client's information in hand, the task of the expert system is to use the same considerations in defining an optimum savings and investment strategy that the human expert uses. The mandate in the system design phase is to convert the expert's decision process into a set of rules and relationships that can be incorporated into the expert system.

All expert systems are goal oriented in that they are trying to solve a problem. In this case the problem faced by both our human expert and our expert system is "How do I structure a plan for this individual that best allows him to meet his financial goals with a minimum of pain?" It is not good enough to propose just a solution that works, we must select a strategy that takes into account a client's fears, goals, loves, hates, desires and more.

5.1 Knowledge Engineering

Knowledge engineering is the process of determining how it is our financial planner makes his decisions when he is preparing a strategy or plan. At this stage two Brant knowledge engineers have met with the financial planners about a dozen times in trying to distill only the rules of thumb that the expert is using.

Before getting into specific examples of the process and the rules it is worth reviewing a few basic "rules" about knowledge engineering.

The first thing that Brant's knowledge engineers discovered was that, unlike a conventional systems analyst role where the end user has too many suggestions and you must weed out the deliverables, our expert had the opposite problem. He appeared to have nothing to say.

Our financial planner explained up front that he didn't really have rules, but rather made his decisions intuitively based on years of experience in "reading" clients personalities. We were informed that there was not a list of rules or procedures that he followed to make his decisions.

This response is actually a well documented "pseudo-truth" that has been addressed in many Expert Systems text books. The problem has been termed the "Expert Paradox", where psychologists have discovered that the better your expert, the less capable he or she will be in describing how they make a decision. It has been determined the experts use what has been termed "compiled knowledge", where over the years the individual rules or conditions in a circumstance become compiled so that the expert actually does not use basic rules but can jump straight to the intuitively correct answer. The knowledge engineer's job is to decompile the knowledge into its composite rules.

Because the expert does not even recognize that he is using compiled knowledge, it is useless to ask him or her "What route did you use in this instance?" The expert honestly may not know what he had to do. What has become fairly standard procedure in this type of process was to provide the financial planning expert with example after example, scenario after scenario, and ask what he would do and why. From these actions or responses our knowledge engineers distilled the necessary rules for the expert system.

5.2 Rules from the Expert

An example of the process and rules from the knowledge engineering stage can be provided based upon one aspect of the decisions that our expert would make; specifically, determining the risk tolerance of the individual or family in respect to their investment portfolio.

In our discussions with the expert on how risk tolerance played into his decisions about portfolio structuring, we asked how he picked a factor. In the "Fact Finder" on the client, the client had been asked to rate his own risk tolerance from 1 to 9, where 1 was extremely conservative (ie: he would only invest in guaranteed investment certificates in major banks that were insured) and 9 was extremely daring (ie: all of his money was in penny stocks of the Vancouver Stock Exchange or with bookies across the country).

The first response from the expert was that he used the number provided by the client to determine risk tolerance. When asked what he did with a similar number provided by the spouse of the client, he informed us that he took that into account as well. We discovered that he looked at such factors as who made the investment decisions, who appeared more knowledgeable about investments, who was the primary breadwinner and several other factors before he selected a risk tolerance that may not relate to either the spouse's or the client's selection.

A similar process occurred when we discussed how the selection was made for the family's after-tax income after retirement. It turned out that the client was asked this question directly, but in almost all cases this number was ignored. The expert asked a wide variety of lifestyle questions that were each broken down into composite numbers. The planner always selected the income requirement figure that was largest, regardless of how it was arrived at. In almost every aspect of the knowledge engineering process we discovered that a myriad of "invisible rules" became apparent once the questioning got underway.

There may be dozens and dozens of rules and relationships that affect the selection of a risk tolerance. From these relationships that were derived from examples or case studies, generalized rules were determined.

risk tolerance (RT) if
client tolerance (RT) and
spouse's tolerance (RT).

In other words, if the client and spouse express similar tolerances, go with it. If the client is also the primary breadwinner, we go with his or her specified tolerance.

6.0 Conclusion

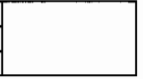
At this stage we have completed the 4GL system and are in the process of refining the expert system. We are pleased with the implementation and excited by the success of our first hybrid system of 4GL and 5GL systems.




HP 3000

INTERNATIONAL CONFERENCE

VIENNA 1987



	HP 3000	AI05/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Artificial Intelligence Applied to the HELP Function

Robert Stanley
Cognos Incorporated
 3755 Riverside Drive
 Ottawa, Ontario
 K1G 3N3
 CANADA
 (613) 738-1440

Traditionally, on-line help for computer software has been based on one of two broad approaches, the on-line manual, and context-sensitive help text. Both mechanisms have spawned numerous sub-genres, many of which demonstrate considerable ingenuity and sophistication. The key factor in the development of these help facilities has been the increasing availability and decreasing cost of high capacity storage, both memory and disk, which has made it feasible to have large volumes of text on-line to computer users.


Most recently, further improvements in the price-performance ratios of computer hardware, coupled with steady progress in artificial intelligence research, have made it possible to consider harnessing so called knowledge-based technologies to the provision of help facilities. Smart help, as it is usually known, can simply be a more sophisticated version of one of the two existing approaches, but this technology also introduces the possibility of new mechanisms.

This paper describes a research project designed to develop a prototype advisor for Cognos' QUIZ report writer. An advisor is a knowledge-based program capable of conducting a dialogue with a user and answering questions within its domain of knowledge, in this case knowledge of a 4th generation computer application programming language. As well as describing the results of the research to date, this paper briefly introduces other potential applications of the technology, discusses their viability, and proposes likely time-scales for their practical availability.

Historical Background

I vividly remember the first time I saw a video display terminal which had a dedicated *help* key; finally, it appeared, a manufacturer was addressing the real needs of computer users. My initial excitement rapidly turned to frustration as I realized that the key didn't actually do anything other than generate a code that could be uniquely identified within a program, *should the programmer choose to check for it*. The experience was a salutary one, in as much as it made me examine some of the fundamental underpinnings of computer systems, which in turn sparked an interest in human-computer interaction that has never waned. But on that day in the early 'seventies, it was the frustration and anger at the thoughtlessness of system designers that predominated. How, I wondered, could an industry continually introduce potentially wonderful mechanisms that for all practical purposes have no useful application?

The answer, as always, turns out to be simply the perpetuation of a tradition that has its origins in the exigencies forced by engineering limitations in pioneering days. The commercial data-processing industry has been, and generally still is, hidebound and conservative in the extreme; only a trickle from the flood of research ever finds general acceptance, and then usually only under the ægis of a determined champion or visionary. The classic example of this in recent years has been the introduction of the Apple Macintosh, which has finally won acceptance for ideas that have been discussed for more than a decade. Of course, the ideas also required the introduction of affordable delivery technologies for their realization, and this is the heart of the problem.

	HP 3000	A105/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

In the earliest days of computing, all users were programmers, and errors manifested themselves as system failures of varying degrees of subtlety. Tracking down the source of a problem was a time-consuming and demanding exercise, and the only help available was discussion with colleagues. Non-programming users first appeared on the scene with the introduction of batch-processed applications, which were accompanied (after the need for them rapidly became apparent) by detailed user instructions for running the application, and a book full of possible error situations, each identified by a cryptic code. Hardware limitations, particularly in memory capacity, necessitated the use of codes rather than text messages. The form of the documentation, which was written by the program authors, reflected the tastes and habits of the programmers, which in turn were shaped by the development environment in which the programmers worked, which ultimately were rooted in the engineering design decisions taken by the hardware manufacturers. The only variation on this theme was provided by the introduction of separate quality assurance or 'testing' sections, who at least verified the correctness of the documentation as well as the applications.

It was really the appearance of interactive computer terminals in the late 'sixties which first started a trend towards addressing more fundamental user needs than simply the raw mechanism for running an application. The problems faced by an unsophisticated computer user (usually described as one with no real understanding of the operating system and programming language characteristics) in attempting to successfully execute a job necessitated impractical levels of training or a radically different approach. In the short term, operators developed startling skills, and major computer centres placed all programmers on operational trouble-shooting detail, frequently on a three-shift basis. The answer was the development of interactive or on-line help facilities, but these were slow in arriving because they required vast (by contemporary standards) amounts of on-line direct-access storage.

The most widely adopted approach was that of context-sensitive help, whereby an interactive user could enter a predetermined key sequence (usually a question mark) at any point where the application was waiting on input. Of course, this only worked where the application design and the programmer had together conspired to provide some meaningful information to be displayed on request. Poor programming all too frequently resulted in a help message of the form:

EMPLOYEE NAME: ? "Enter the name of the employee"

when what the user really wanted to know was whether the initials were supposed to precede or follow the name, or if the name was supposed to match some other field, and so on. However, this system has merits, and is widely in use today, usually with multiple levels of information being made available. I.e., one request returns possible syntax of an entry, a second the value restrictions, a third a textual description of the purpose of the field. The major limitation with this approach is that the user is utterly dependent on what the program developer has decided is relevant, and has no alternative recourse except to consult any available written documentation, or to contact the program developer.

The other approach that has been widely adopted is the concept of the on-line manual, whereby the user is free to browse through a structured set of documentation made available on-line. This mechanism was slow to be adopted because of its massive requirement for on-line direct-access storage. Now that disk storage is cheap, comparatively speaking, this approach is being more widely adopted, particularly because it places little or no demand on the programmer. The on-line manual is typically accessed via one simple mechanism, and then behaves as a self-contained application. However, from the user's point of view, the on-line manual suffers from all the deficiencies of paper documentation: obscurity, poor layout, useless or non-existent indexes and/or tables of contents, inconsistency, and omissions. An interesting recent study¹ also revealed that many users fail to

¹ "Helping Users Find Help: Models of Online Help Organization"; Marjorie S. Horton, IBM Human Factors Center, in the ACM/SIGCHI bulletin for October 1986.

exploit the capabilities of even the best designed on-line information systems, because they fail to perceive the structure of the on-line information, even when it is designed to be *intuitively* obvious. Cognos' products to date have all followed the approach of providing multi-layered, context-sensitive help facilities. However, the technical writing group have recently produced a working prototype of an on-line documentation system, which includes 'smart' indexing and searching, as well as editable examples that can be executed against a demonstration database from within the help context. Also, the research division has now produced the first working prototype of a completely different kind of on-line help, an advisory system based on artificial intelligence techniques. It is this 'advisor' that is the subject of the remainder of this paper.

Overview

The aim of the 'Advisor' project is to create a software advisor or assistant capable of answering questions, phrased in a minimally constrained English-like language, about the use of a typical Fourth Generation Language (4GL). For the prototypes we have chosen QUIZ, Cognos' 4GL report writer, as the experimental subject, both because it is a mature and reasonably well understood product, and because there is a large body of users with a well documented history of problems encountered using the product.

The 'Advisor' is a joint industrial-academic project between Cognos Incorporated and the University of Ottawa, with grants from the National Research Council of Canada, The Natural Sciences and Engineering Research Council of Canada, and the Ontario Ministry of Colleges and Universities. The work is being carried out at the University of Ottawa's Artificial Intelligence laboratory, by a mixed team of Cognos research group members, and University of Ottawa professors, graduate students and paid assistants.

The project was started in the Autumn of 1985, and is scheduled to be completed in the late Summer of 1987 with delivery of a full demonstration prototype system. The first year of the project was spent on theoretical research, with practical experiment starting early in the Summer of 1986. A proof of concept prototype was completed at the end of October 1986, and successfully demonstrated to the various sponsoring organizations in the following month.

The current development uses two different software environments running on two different computer systems, linked by a 10 megabit/sec Ethernet: Quintus Prolog running on a Sun/3 workstation, and IntelliCorp's KEE™ (Knowledge Engineering Environment) running on a Xerox 1186 (Dove) InterLisp machine. This separation was deliberately chosen to ensure that the research project investigated the relative merits of a number of different technical approaches, although experience to date indicates that we might have encountered severe difficulties in attempting to build the complete system to run on a single machine of the classes available to us.

It is worth noting that at our current stage of experiment enormous computing resources are required to implement our various mechanisms. Only when we have the full prototype completed will we start to address the issue of efficient implementation in a less specialized environment. Our ultimate goal is to produce a system capable of running on an IBM PC/AT, or equivalent. Preliminary experiments in this direction indicate that this should be an achievable goal within a very few years, partly because of the appearance in this market of sophisticated AI tools, and partly because of the continuing trend in increasing hardware performance for a given price. It was only the comparatively recent availability of affordable, very powerful work-stations, epitomized by the Xerox Lisp machine, that made this project possible.

Approach

We adopted a somewhat unusual approach as we embarked on the research project by consulting the records of Cognos' telesupport group for documentary evidence of problems encountered by QUIZ

users. The important point was that these were end users of QUIZ, who were attempting to use it as the tool of choice to solve real-life data-processing problems. Their calls to the telesupport group could thus be construed to be typical of questions that any 'advisor' should be capable of answering. We screened the questions for obvious irrelevancies, and then classified them according to a system which we developed. The majority of the questions were split between only two categories, the 'how do I ...' (HDI) and the 'why did ...' (WHY) or causal; other categories, including syntactic (SYN), explanation of obscure error messages (ERR), definitional (DEF), and hypothetical (HYP) accounting for the rest. The HDI's alone accounted for more than half the questions, and it was decided to make answering this type of question one major focus of the project. A second sub-project was initiated to tackle the next two most frequent categories, the WHY's and the SYN's. The SYN's (questions such as "can there be more than one ACCESS statement in a report?" or "what is the syntax of a FOOTING statement?") are the traditional meat of context sensitive help systems, but accounted for less than 15% of the total number of questions, indicating that what was needed did indeed lie outside the capabilities of contemporary help systems.

Once we had classified the questions, we made a detailed study of those falling into the categories of interest, further analysing them by topic. Perhaps the greatest beneficiaries of this exercise were our technical writers, who received a weighted list of six topics that clearly gave many users difficulties, and which therefore required new treatment in the reference manuals. Once we had classified the questions by topic, decided what problems appeared to characterize each, and what general characteristics could be inferred, we performed a final screening of the questions to eliminate those which could be classed as relating to very obscure issues or to technical tricks outside the mainstream use of QUIZ. This left us with a representative set of some 200 questions, and we defined our goal for the working prototype as the capability to answer any question which can clearly be shown to be directly related to one of these 200 questions.

Of course, in order to prove the correctness of an answer, one needs at least a model answer for each of the questions to act as a yardstick. We therefore circulated our 200-question set to members of various groups within the Cognos head office, including tele-support (where the questions originated), product development, and a variety of technical specialists. Interestingly enough, in many cases there proved to be no one correct answer, indicating the depth of the problem when using one of the sophisticated software tools that characterize a 4GL. Where more than one answer was produced for any given question, we had the experts weight each answer and recorded the set of answers, ordered by weight, with the question.

At this point we had the necessary information to start designing possible question answering mechanisms, and were able to embark on the real work of the project. We eventually broke the project down into four sub projects, and treated each as a relatively independent development up to the first (proof of concept) prototype stage (P1). These four sub-projects were:

- a parser capable of syntactically analysing any question asked by the user of the system, couched in a formalized English-like language, and integrated into a user interface;
- an HDI question-answering system;
- a causal (WHY and SYN) question-answering system; and
- an 'apprentice' system with the capability of learning from the user.

Each of these is briefly described in the following sections of the paper.

The Parser and User Interface

The parser was developed as a Master's thesis under the guidance of a natural language specialist (Dr. Stan Szpakowicz) at the University of Ottawa. The parser is designed to accept four types of input, as follows:

- an assumption or descriptive text fragment providing background information which may help to clarify the context of a question. These are entered in a restricted form of English (see below), delimited by a full stop, and can be annotated with various special symbols to help the parser. Examples of such assumptions are:
 - 'employees' is a keyed file .
 - 'emp_num' is a key of 'employees' .
 - 'surname' is a key of 'employees' .
- a question, which is also entered in restricted English, but delimited by a question mark. Questions can also be annotated with various special symbols to help the parser. An example of a typical question (in the context of the previous assumptions) is:
 - how do i access 'employees' using 'emp_num' and 'surname' ?
 another example (showing some annotation to help the parser correctly identify dependent clauses) is:
 - how do i print an item at [the beginning of a detail line] ?
- a fragment of QUIZ code, presumed relevant to the questions being asked. Only minimal syntax checking is performed on the code fragment.
- the literal text of an actual QUIZ error message about which questions are being asked.

The subset of English that the parser is designed to accept is in fact a formalism specifically designed for knowledge representation called LESK: Language for Exactly Stating Knowledge. The LESK language has been developed over a number of years by Dr. Skuce of the University of Ottawa, and the work that had already gone into the specification of this language was one of the starting points for the Advisor project. In practice, LESK had to be considerably extended to make it fit our particular application, but its straightforward simplicity is easy to pick up, and once mastered, LESK can be used to express a wide variety of concepts. Any limitation in vocabulary is a relatively trivial problem to solve, requiring only additional entries in the lexicon; the grammar is sufficiently extensive to cope with the variety of constructions needed to state assumptions and frame questions about programming with QUIZ.

A question on its own, or a question accompanied by one or more assumptions, and/or a fragment of QUIZ code, and/or a QUIZ error message constitute a logical query, which is the unit of input to one of the question-answering systems. The parser outputs a successfully parsed fragment in the form of a parse tree, represented either as a Prolog list or as a Lisp S-expression according to a user specification. In the case of the parser failing to parse an input fragment, an interactive dialogue is initiated with the user to attempt to resolve the problem. The user is given the choice of respecifying the fragment (abort parse for this fragment), either ignoring or altering the specific word over which the parser has stumbled, or of undertaking a more technical dialogue via which a new word can be defined in the parser's lexicon.

The parser itself is written in Quintus Prolog, and runs under UNIX on a Sun/3 workstation. The design of the current implementation is based on some fairly standard natural language processing theories for syntactic analysis, driven by a sophisticated lexicon, and is capable of parsing any input fragment in between 0.1 and 0.5 seconds.

Building the lexicon proved to be one of the most interesting exercises, in as much as we wrote a special concordance program through which we fed a large body of machine-readable documentation, including the complete text of the QUIZ reference manual. The output from this gave us some 1,500 words for inclusion in the lexicon, together with all the necessary references and annotations required to complete each lexical entry². This linguistic analysis of existing documentation raised a number of

² A lexical entry is similar to the entry for a word in an English language dictionary, but with a variety of annotations describing the part of speech, and possible rôles that the word can play.

interesting side issues, chief among which was that, while natural language theory suggested that we might expect some 400-450 different verbs to have been used, in fact the total was only about 150. When we investigated this with the technical writing group, we discovered that the technicians in the development group who had vetted the documentation for technical accuracy had insisted on particular usage of a variety of technical terms, and it was this insistence that had led to the relative paucity of the vocabulary.

Much more interestingly, because each of these words was serving multiple duty, they tended to gain a variety of context-dependent meanings, which turned out to be the root cause of a number of the misunderstandings demonstrated in our sample questions. Since the QUIZ documentation has won awards as the best documentation of its kind, this is clearly not an isolated problem, and indicates not only the need for early involvement of technical writers in the product development cycle, but also that formal technical glossaries must be developed which assign unique and unambiguous meanings to each technical term. Interestingly enough, one possible application of the Advisor prototype is to the development and formalization of such technical glossaries for future products.

The parser development thus proved to be an interesting exercise, and the working prototype has proven to be fast and effective in use. It does constrain the user somewhat with respect to freedom of expression on input, but experiment and constant usage over several months have shown that it is easy to adapt to the parser's constraints, and that these do not interfere in any important way with the user's freedom of expression.

The HDI Question-Answering System

The How-Do-I question-answering system (HDI) is the area into which the most effort has been directed, mainly because we wished to explore the capabilities of available large-scale AI tools. For a variety of reasons, we eventually settled on the Knowledge Engineering Environment (KEE™, from IntelliCorp™) running under Interlisp-D on the Xerox-1186 (Dove) Lisp machine. This has proved to be a nearly ideal prototyping environment, but its tremendous richness resulted in a long and relatively steep learning curve which we believe will take some 12 months to travel to the level of neo-expert.

Using these tools, we have developed a question-answering system that utilizes three fundamental mechanisms: frame-based representation of knowledge; a forward-chaining, rule-driven inferential system; and object-oriented methods written in InterLisp. These three mechanisms are combined in various proportions to answer a submitted query in three stages, as follows:

- Using an approach known as semantic interpretation, the parsed S-expression for a user query is received over the Ethernet, and translated into a formal representation using KEE frames, based on the system's *understanding* of the query. Understanding, in the context of semantic interpretation, means being able to classify all phrases of the input query in terms of concepts defined in the knowledge base, and correctly assigning all objects into rôles which fit the activities mentioned.

The semantic interpretation phase can fail completely (I don't understand the question) or may start an interactive dialogue with the user to clarify and disambiguate phrases of the input query which are causing confusion.

- Once the semantic interpreter has completely transformed the query into a conceptual representation as a KEE knowledge base, it invokes a mechanism that attempts to match the query against known knowledge. This matching mechanism may yield a perfect match (the query is completely answerable), a partial match (in which case the user will be solicited to supply additional information), or a mis-match (unable to answer the query). All queries must eventually reduce either to an acceptable match, or a rejection of the query as unanswerable in its submitted form.

An acceptable match is comprised of a set of potentially useful KEE units, each of which is deemed to hold information relevant to answering the query.

- In the case of an acceptable match, a syntax generating mechanism is invoked, which assembles and displays an answer from information held in the set of potentially useful units. This mechanism has sufficient knowledge of QUIZ syntax to enable it to formulate an answer in the form of QUIZ statements, although the resultant code may have instantiation points representing generalities (e.g. <report item>) and unknown specifics (e.g. <file name>); these are represented in angle brackets.

In addition, or as an alternative, to the QUIZ code generated (the definition of an HDI question is one for which an adequate answer is a piece of QUIZ [pseudo-]code), the answerer may generate a plain text message. Typically, this occurs in the cases where the query has an explicit negative answer, e.g. "It is not possible to report a heading to a subfile".

In fact there is no clear boundary between these three stages, and the decision as to which mechanism should be responsible for what has tended to be taken on the basis of implementational ease. However, it helps to think of the process broken down in this fashion, because the major consideration is the construction of generalized mechanisms driven from separate knowledge bases. The more domain-specific (QUIZ in our case) knowledge there is hard coded within the actual mechanisms, the less useful (read: extensible, portable, maintainable, etc.) the system becomes. A brief example may serve to highlight some of these ideas.

The previously discussed query:

```
'employees' is a keyed file .
'emp_num' is a key of 'employees' .
'surname' is a key of 'employees' .
how do i access 'employees' using 'emp_num' and 'surname' ?
```

Is translated by the parser into the following set of Lisp S-expressions, which are combined into a single list before submission to the question-answering mechanisms:

```
Assertion 1: (is_a yes (variable employees) (count_nounphrase 1 ()
keyed_file (variable employees) ()))
Assertion 2: (is_a yes (variable emp_num) (count_nounphrase 1 () key
(variable emp_num) ( (of (count_nounphrase 1 () keyed_file
(variable employees) ())))))
Assertion 3: (is_a yes (variable surname) (count_nounphrase 1 () key
(variable surname) ( (of (count_nounphrase 1 () keyed_file
(variable employees) ())))))
Question: (hdi yes () access ((nil sub (personal_pronoun i)) (nil d_o
(count_nounphrase 1 () keyed_file (variable employees) ()))
(nil using (and (count_nounphrase 1 () key (variable
emp_num) ((of (count_nounphrase 1 () keyed_file (variable
employees) ()))))) (count_nounphrase 1 () key (variable
surname) ((of (count_nounphrase 1 () keyed_file (variable
employees) ()))))))
```

Careful examination of the above S-expressions reveals that all the information represented in the assertions is duplicated in the question. In fact it would be possible to generate an identical question S-expression by posing the following question without any assertions:

```
how do i access a keyed file 'employees' using a key 'emp_num'
of 'employees' and a key 'surname' of 'employees' ?
```

LESK allows the user considerable flexibility and freedom of expression when formulating queries. The Semantic Interpreter takes the S-expression list output by the parser, and creates a knowledge base containing the following units:

- 01 an ACCESSING activity, with a target of 02, and an agent of 03;
- 02 a FILE object, named 'employees', with an access-mechanism of 'keyed', a key-list of 04 and 05, and membership in the class of DATA-FILES;
- 03 a set-of KEYS, pointing to 04 and 05;
- 04 a KEY object, named 'surname', with membership in the class of DATA-ACCESS-KEYS;
- 05 a KEY object, named 'emp_num', with membership in the class of DATA-ACCESS-KEYS.

A number of inferences have been made, including the fact that the keys are data access keys rather than sort keys, and that the file is a keyed *data* file. In this latter case, while the 'keyed' attribute can be readily inferred even when it is not explicitly stated, the 'data file' attribute is not obvious (there are a number of possible alternatives). Where disambiguation proves difficult, the Semantic Interpreter either asks for confirmation, e.g.

Please confirm that the referent: FILE (employees) is a member of the class of DATA-FILES (Y)/N ?


or leaves the problem to be tackled by a subsequent processing phase. At present the Semantic Interpreter draws the inference and requests user confirmation, but this has caused problems in more complex situations, and we may choose to let this type of problem stay unresolved until a subsequent phase.

The second (matching) phase extends this first representation, by altering existing units and generating new units, to generate the following set of potentially useful units:

- 01 an ACCESSING activity, with a target of 02, and a sub-activity of 06;
- 06 a LINKING activity, with a target of 07 and 08, and an agent of 03;
- 02 a FILE object, named 'employees', with membership in the class of PRIMARY-DATA-FILES;
- 07 a FILE object, named 'employees', with an access-mechanism of 'keyed', a key-list of 04, an alias of '<alias-for-surname>', and membership in the class of SECONDARY-DATA-FILES;
- 08 a FILE object, named 'employees', with an access-mechanism of 'keyed', a key-list of 05, an alias of '<alias-for-emp_num>', and membership in the class of SECONDARY-DATA-FILES;
- 03 a set-of KEYS, pointing to 04 and 05;
- 04 a KEY object, named 'surname', with membership in the class of DATA-ACCESS-KEYS;
- 05 a KEY object, named 'emp_num', with membership in the class of DATA-ACCESS-KEYS.

Notice that the name of an object is merely another attribute of a unit and not its identifier, thus allowing three FILE units to exist for the same file. At this point the query can be classed as fully matched, because all the concepts mentioned have been successfully matched with known concepts, with sufficient precision to generate an answer. The answer generating phase is invoked, and in fact increases the set of potentially useful units with a number of units to represent syntactic fragments. The new units are:

- 09 an ACCESS-CLAUSE, with a target of 02, and a link-list of 10 and 11;
- 10 a LINK_CLAUSE, with a target of 07, and a linkage-spec of 12;
- 11 a LINK_CLAUSE, with a target of 08, and a linkage-spec of 13;
- 12 a LINK-BY-KEY-SPEC, with a file-ref of 07, and a key-ref of 04;
- 13 a LINK-BY-KEY-SPEC, with a file-ref of 08, and a key-ref of 05.

	HP 3000	A105/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

These are much simplified representations, but show the type of cross-references that are constructed, together with a degree of redundancy that eases navigation between units in the knowledge base. From these syntactic fragments, the following QUIZ pseudo-code is generated and displayed in the answer window:

```
ACCESS EMPLOYEES &
LINK TO SURNAME OF EMPLOYEES ALIAS <alias-for-surname> &
LINK TO EMP_NUM OF EMPLOYEES ALIAS <alias-for-emp_num>
```

This is the correct answer to the query, but will require the user to code the actual alias names to be used before the statement becomes executable.

The P1 prototype is capable of answering a handful of queries on the topics of accessing (the previous example is a typical such query), reporting and sorting. Achieving this required knowledge representation for a total of some 85 concepts (both objects and activities) plus about a dozen syntactic fragment specifications, supported by some 70 rules and several dozen Lisp methods. However, the fundamental mechanisms appear readily extensible, and adding knowledge about new concepts should prove merely time-consuming.

The Causal Question-Answering System

The causal question answering system, known as QAUZ, is the subject of a Ph.D. thesis by Branka Tauzovich, a member of Cognos' Research Division on scholarship at the University of Ottawa. The requirements for originality in a doctoral dissertation necessitated keeping this portion of the project independent, and it has thus been treated as a parallel development. The QAUZ system has been implemented entirely in Quintus Prolog on a Sun/3 workstation; its only direct interaction with the rest of the Advisor is that it invokes the same parser to process its raw input.

The QAUZ system typically accepts a fragment of QUIZ code as part of a query, because the usual WHY question requires the code in order to be able to establish the context in which the query is being asked. For ERR type questions (explaining error messages), should it prove impossible to answer the question without reference to the context, the user will be prompted to enter the code. In the prototypical implementation no provision was made to allow direct access to machine readable QUIZ code, but this would certainly be the approach adopted for any more serious implementation. The text of QUIZ error messages are also acceptable as input, because one major category of user question is the "why did such and such an error occur?".

The QAUZ system comprises a knowledge base (implemented mainly in declarative rules, but augmented with a custom frame-based mechanism), which describes a mini-QUIZ model (a subset of QUIZ similar to that used in other parts of the Advisor project), a user interface with links to the parser, a forward-chaining inference mechanisms, a backward-chaining inference mechanisms, a rule base, and a variety of tools such as frame compilers. The system uses a combined forward- and backward-chaining strategy to reason forward from a question, and backward from actual QUIZ code, to generate an explanation which satisfactorily accounts for the reported facts.

While processing a problem, the QAUZ system constructs an internal representation of the problem state which shows all the various dependencies and causal mechanisms that may be contributory factors. Even if a focused answer cannot be reached by the system, a browsable display of the internal representation of the problem frequently yields sufficient information to enable the user to answer the query. The following example shows the QAUZ system reasoning about the circumstances under which a default heading should appear on a report; note that the system is capable of handling inconsistencies:

```
Enter your QUIZ code:
>> REPORT X HEADING "Name"
>>
```

> Why is the column heading set to the dictionary heading?

Your query is INCONSISTENT!

Contradiction found regarding:
the column_heading of a report_item is set to the
dictionary_heading of an item

Either your QUIZ statements are incorrect
or you did not formulate your question properly!

Assuming the latter...

The NEGATION of your question is TRUE
because the following DOES NOT HOLD:

- all record items are requested for reporting (f),
CONSEQUENCE OF:
using 'REPORT with the ALL_option' statement (f)
- AND
- a record_item is an accessible record_item (?),
CONSEQUENCE OF:
the record_item is a declared record_item (?),
 - the record_item is a record_item of a file (?)
 - AND
 - the file is a declared file (?),
CONSEQUENCE OF:
using first_file_declaration
in 'ACCESS' statement (?)
 - OR
 - using subsequent_file_declaration
in LINK_TO_part
in 'ACCESS with a LINK_TO_option'
statement (?)
- OR
- a default for the column_heading of a report_item
is used (f),
CONSEQUENCE OF:
using format_part without HEADING_part
in report_item
in report_items
in report_group
in 'REPORT with a report_group' statement (f)

> Why is not the column heading set to the dictionary heading?

Because the following DOES NOT HOLD:

- all record_items are requested for reporting (f),
CONSEQUENCE OF:
using 'REPORT with the ALL_option' statement (f)
- AND
- a record_item is an accessible record_item (?),
CONSEQUENCE OF:
the record_item is a declared record_item (?),
CONSEQUENCE OF:
the record_item is a record_item of a file (?)
- AND
- the file is a declared file (?),



```
CONSEQUENCE OF:
  using first_file_declaration
  in 'ACCESS' statement (?)
OR
  using subsequent_file_declaration
  in LINK_TO_part
```

OR

a default for the column_heading of a report_item
is used (f),

CONSEQUENCE OF:

```
  using format_part without HEADING_part
  in report_item
  in report_items
  in report_group
  in 'REPORT with a report_group' statement (f)
```

The fundamental mechanisms of the QAUZ system are capable of dealing with all the various categories of question discussed in the Approach section of this paper. However, only the WHY, ERR, HYP, and SYN categories have actually been addressed in detail, and these only for the defined mini-QUIZ subset of QUIZ.

Obviously, the QAUZ and HDI systems overlap in capability, but the need to keep the QAUZ system independent of the rest of the project has allowed us the luxury of developing two totally separate approaches. The second prototype (P2) will benefit from the work done in both systems.

The QUIZ Apprentice

The QUIZ Apprentice is a parallel project being undertaken at the University of Ottawa by a team led by Dr. Stan Matwin, and is not formally a part of the Advisor Project. However, it sprang from work started by Dr. Matwin in the early stages of the Advisor Project, and there are close ties between the two teams.

One of the chief reasons for the separation is that the Apprentice has been implemented entirely in Prolog, and as yet we have found no way to tightly couple Prolog with a Lisp-based knowledge representation mechanism. The Apprentice thus has its own internal model of QUIZ, complete with its own knowledge base constructed entirely in Prolog. In its earliest incarnation it used approximately the same subset of QUIZ as was chosen for the HDI question-answerer.

The prime characteristic of the apprentice approach is that it is capable of learning. For a rule-based system, this means that the operational system can create new rules as a standard part of its normal processing, as well as extending its knowledge with each new solution generated. In practice, the system designer seeds the knowledge base with a basic set of rules, using an interactive teaching mechanism to do this. This infant system can then be introduced into a user environment. Whenever it encounters a new problem for which it is unable to find a solution, or when it offers an incorrect solution, the user can enter the interactive teaching mode and help the system define a set of rules which, in conjunction with its existing knowledge, will enable it to generate the correct solution.

The system's knowledge is based on problem-solution pairs, and all that is needed to teach the system is to supply the correct solution to a new problem. The apprentice not only stores the problem-solution pairs but is capable of reasoning, so that by decomposing problems into partial problems, and matching these against its knowledge base, it is capable of solving problems for which no explicit solution is stored. Although the underlying mechanisms are conceptually simple, even the initial prototype QUIZ Apprentice was capable of demonstrating remarkable performance.

Clearly the first user of an infant system needs to be fairly knowledgeable, in order to detect the situation where incorrect solutions are being offered by the apprentice. However, experiment has shown that the number of rules needed to adequately cover a given topic, while not fixed, can be quickly identified. In operation, the Apprentice rapidly builds up a few dozen rules (typically 2- or 3-) for a given topic; thereafter, new rules only need to be generated to cover exceptional cases. As soon as this point (where the rate of rule creation drops off sharply) is reached, the system can be turned over to a naïve user. Typically, from this point on there will be no incorrect solutions generated, and the system will only require teaching when new problems are encountered. When this occurs, the user can ask an expert to supply the correct solution.

The advantage of this approach is that it eliminates the problem of requiring a technical specialist to make changes in the system's stored knowledge, which is the major problem with most of today's operational expert systems. However, the more complex the fundamental knowledge representation and reasoning mechanisms, the harder it is to ensure that the rule construction mechanisms are correct: and the need to detect potentially incorrect solutions, at least in the early stages, introduces the much wider issue of the verification of knowledge bases. Not for nothing is this approach named 'apprentice', and it is important to remember that the step from apprentice to journeyman has always been a big one, requiring years of effort carefully directed by an experienced and capable master.

The First Prototype (P1)


The first prototype (P1) was completed in October of 1986, and was successfully demonstrated to Cognos' Research management and to a committee from the National Research Council in mid November. The P1 system requires a Sun/3 workstation with at least 4 megabytes of real memory running under UNIX (System 4.2) with Quintus Prolog, linked via a 10 megabit Ethernet to a Xerox 1186 (Dove) Lisp machine with the maximum memory configuration (3.7 megabytes) running under Interlisp-D with IntelliCorp's KEE™ version 2.1. Both systems require access to at least 50 megabytes of individual disk space.

For P1, the various components of the system were linked together within a very basic user interface, written in a mixture of C and Quintus Prolog, and running on the Sun workstation. This user interface initializes all the communications mechanisms, controls the Lisp machine as a slave AI problem solver, and allows the user to enter, edit, debug, and submit (to the appropriate question-answering system) all the various components of a query. Although primitive in appearance and behaviour, this interface has proven to be a robust and indispensable tool for working with all the various components of the system, as well as acting as a prototype for the more ambitious interface planned for the second prototype (P2).

This system could be run entirely from the Xerox Lisp machine (the slave!) by opening an Interlisp window and running a terminal emulation via which the user interface on the Sun could be accessed. Queries could thus be entered in this window, and the rest of the screen used to display the question answering process in action.

For P1 we limited ourselves to a handful of questions on a few major QUIZ topics, namely: data access, reporting, sorting, and selecting (QUAZ only). A set of mechanisms that could be adapted to all these topics were developed, and based on the experience with P1 we expect little difficulty in extending the knowledge base to include new topics, as well as to cover these first topics more extensively. QUIZ has some 31 main verbs, and there is a rough equivalence between these and Advisor topics. The main purpose of the P1 prototype was to demonstrate the feasibility of answering questions, and to test a variety of implementation mechanisms. Both of these objectives were satisfied.

Although performance was not an issue for this prototype, it may be of interest to note that it took the KEE/Interlisp-based mechanisms between 4 and 20 seconds to answer each P1 question. This in a

	HP 3000	A105/13
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

development environment with multiple levels of debugging active, and with extensive graphic traces being displayed. One of the P2 issues that we intend to address is the question of what sort of response might be considered reasonable for an advisory system.

The QAUZ system P1 prototype runs totally independently, and is self contained on a Sun/3.

The Second Prototype (P2)

The second prototype (P2) is scheduled for July 1987, and will be based on the work done to date, once this has been extensively reviewed. At present it is too early to state with any certainty what will and will not be attempted, but a number of possibilities are open. These include:

- Integrating the entire system into a single machine environment. This will require the availability of new tools, notably KEE on the Sun/3.
- Coping with procedural or time-dependent problems.
- Extending the P1 prototype with lots of knowledge to give coverage of the majority of QUIZ rather than the limited subset used for P1.
- Developing a sophisticated user interface, taking full advantage of windows, interactive graphics, and other available techniques.
- Closely coupling Prolog with Lisp-based knowledge representation mechanisms.
- Transferring the existing system into a smaller self-contained environment running on a computer such as the IBM-PC/AT.
- Replacing the current independent syntactic parser with extensions to the semantic interpreter.
- Development of deeper models of QUIZ, to enable greater understanding of queries by the system, and thus generation of more sophisticated answers.
- Focusing on a single topic, such as file access, and attempt to create an advisor with effectively complete understanding of this one topic.
- Development of more sophisticated reasoning strategies, capable of dealing with incomplete queries.
- Development of alternate answering mechanisms, to enable the user to get more out of the system than one blunt answer.

This last problem is really the heart of the design problem. It is very easy to develop a system that answers a question of the form 'Can you tell me the time?' with a blunt 'yes'; this is of little use, and serves only to irritate the user. It is essential that an advisory system be capable of interpreting the reason behind a question that is asked, and thus generating a meaningful answer, rather than blindly answering the literal interpretation of the question that was asked.

It used to be that the way to test whether a computer system was working was to pose the same problem repeatedly, and check that the answers were consistent, which usually meant identical. Perhaps *the* distinguishing feature of a knowledge-based system is its ability to learn. This means that the second time a question is posed the response should at the very least be faster, and that multiple repetitions should trigger some alternate action. In an ideal system this would include proposing alternate approaches in an attempt to help the user reach a solution to the real as opposed to the expressed problem.

Given the limited lifespan of the current project, it is unrealistic to expect that we will solve more than a fraction of these problems. However, it is to be hoped that we will make some forward progress, and perhaps come closer to the currently perceived ideals.

Practical Implementations

Thus far this paper has been concerned mainly with reportage of what has been achieved, and a description of possible short-term research goals. Quite clearly, the P1 and P2 prototypes have little direct application in the resource-conscious practical data-processing world, relying as they do on costly and largely non-standard dedicated hardware and software environments. However, although the goal of research is knowledge, there are some clear pointers to what the future holds in store.

Firstly, these technologies are practical. That is to say, it is possible to build a knowledge-based system capable of doing real work in real environments. The problem is that at present it costs hundreds of thousands of dollars to implement such systems, which makes them cost-effective only in the most specialized of applications. Most of this cost is sucked up by the need for highly skilled development personnel, expensive tools, and the fact that very large machine resources are typically consumed by the operation of such systems.

Today, machine costs are falling, and the latest generation of chips (DEC MicroVax, Intel 80386, Motorola 68020, National Semiconductor 32032, TI Explorer, etc.) offer more than adequate raw power. Once this power has been packaged in suitable architectures, it will become an easy matter to boost a conventional system with an AI co-processor, much as floating-point co-processors have been integrated into conventional architectures. In addition, the cost of stand-alone micro-systems will continue to fall, at least in terms of their price/performance ratios. The machine resource problem will disappear as certainly as has the memory problem over the last few years.


Today's costly tools are rapidly being migrated from the esoteric Lisp machine environments of their engendering to the relatively conventional UNIX world, and will soon be available implemented in conventional programming languages such as C. At the same time, the primitive first generation PC-based AI tools are giving way to sophisticated second and third generation offerings that are beginning to rival the big tools. History tells us that competition will trigger falling prices until levels deemed acceptable by the user community are reached.

The problem of skilled personnel for development will also diminish as more and more sophisticated tools are developed. A logical extension of the QUIZ Apprentice is a system that can be taught by a domain expert who can be relatively ignorant of the internal mechanisms. This problem is analogous to the problem of data-base specialists, who are still required in sophisticated environments, but who can be ignored when simple applications are implemented using sophisticated DBMS systems.

Knowledge-based systems are a reality. Within three to five years they will become the norm. Should a QUIZ advisor be considered a reasonable product, a commercially attractive implementation capable of running on a dedicated IBM-PC/AT or equivalent could be brought to market within three years, i.e. first quarter 1989. It is by no means clear that such a product has a market, but the underlying technology has many applications, and a few of these are introduced in the following paragraphs.

Although true advisors are a possibility, they are unlikely to appear integrated into applications (in the same fashion as current help facilities) for several years yet. However, they are feasible packaged as stand-alone applications on dedicated hardware, which certainly introduces the possibility of a PC-based system. An alternative solution is to build a multi-user, server-type front end for a large-scale, central system dedicated to running one or more advisors, and allow users to query an advisor remotely via local network or telecommunications links.

A much more likely scenario is that, instead of an advisor, a teacher will be created. Computer-based training is a growing field, as the widespread adoption of computers, particularly at the desk-top level, exacerbates the problem of the relative scarcity of teachers. The AI community is currently focusing considerable attention on the problems of teaching, and AI is seen as the most likely technology to provide a quantum improvement over the rather pedestrian on-line tutorials available today. The key to a good teaching system is not its ability to detect correct answers, but its capacity

	HP 3000	A105/15
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

for understanding *why* a wrong answer has been entered, and for developing a strategy to correct the user's misapprehension. The same knowledge required to implement an advisor serves to implement a teacher, although in the case of the latter this must be augmented with knowledge about the types of mistakes that can be made. Typically the ratio of information is about 8:1; for every correct fact there are roughly eight possible misunderstandings.

One direct application is to a tool to help software product designers keep their facts straight, and ensure that all the various components are functionally complete and correctly interconnected. This includes the development of technical glossaries in which each term is uniquely and unambiguously defined. Although such a system is a design tool, aimed at ensuring the correctness and completeness of a software product, it could also be extended to create outline technical documentation that conforms to a standard structure. By the same token, such a system could also ultimately be used to proof final documentation for completeness, accuracy, and consistency. The key to making any such system work will be ensuring that it is sufficiently easy to use, and directly beneficial, that development teams will be careful to keep its knowledge current and accurate. The major flaw in most paper-based design tools has been that the effort required to keep the paper up to date has tended to detract from the true development tasks.

Beyond such design aids, which ought to be equally useful to all builders of software applications, lie some very exciting possibilities. It is only a short step from an advisor which answers a 'how do I' question with a piece of pseudo-code to a system capable of generating executable code. This would allow users to program a system at a very high level, by expressing their needs in a fairly informal fashion. With an adequate control structure, the AI-based programming tool could help the users refine their ideas by posing questions until the application was completely defined. The major step forward needed to make this sort of tool a reality is the development of knowledge bases describing typical applications, but it transpires that a significant portion of the knowledge needed to implement an advisor is just this.

The key to all these systems lies in the user interface, which must allow the user to communicate with the system in a natural fashion that is as easy to use as the natural language and scribbled diagrams we use to communicate our ideas to other humans. The human tendency to make life as easy as possible means that we seldom use a long form when a shorter is available; typing true English at a keyboard is a non-starter, if quicker mechanisms are available. Even if affordable voice input technology makes spoken natural language a possibility, we will still need the ability to diagram, annotate, doodle, and so on. It is the availability of these latter mechanisms that will determine the time-scales for the widescale introduction of true AI-based tools.

Within two years, high resolution (1276 x 1024 minimum) colour graphic displays will be achieving affordable levels. We already have excellent interactive pointing devices. Three years will see the arrival of reliable voice input technology; it is already affordable, but has too low an accuracy rate to be really useful. In the same timescale very large data storage (optical disks), high quality hard copy devices (laser printers), and very fast desk-top architectures will all have achieved widescale commercial viability. All that is missing is the software, first its development and then its acceptance into the market place. Today, it exists only in fragmented prototypical form in research establishments around the globe.

A useful model that we can examine for what to expect is the introduction of the graphic desk-top metaphor pioneered by Xerox on its Star systems. Although demonstrably viable, it was far too expensive for all but a handful of users, and even the far more affordable Apple Lisa failed to achieve widespread acceptance. The introduction of the Apple Macintosh, however, offered the necessary price break-through that triggered both widespread acceptance of the technology, and a trend towards making an interactive, iconographic interface standard on all systems. The Macintosh is not yet four years old, at least in terms of its market availability. By analogy, we can expect today's high priced experimental interfaces to reach the mass market level in the same timescale as the necessary hardware



becomes affordable. Another two to three years should serve for such products to gain credence similar to that now afforded the Macintosh. Thus, by 1992 it is very probable that systems based on current experimental advisor technology will be commonplace.

Acknowledgements

To the National Research Council of Canada, the Natural Sciences and Engineering Research Council of Canada, and the Ontario Ministry of Colleges and Universities for various grants which have helped make this project possible.

To the University of Ottawa for its joint sponsorship of this project.

To Drs. Skuce, Matwin, and Szpakowicz of the University of Ottawa, and Dr. Oppacher of Carleton University for their participation in this project, and for advice freely given.

To Branka Tausovich, Charles Truscott, Sylvain Delisle, Deborah Lazar, Patrick Constant, Yannick Toussaint, and Claude Queant for their theoretical and practical technical contributions.

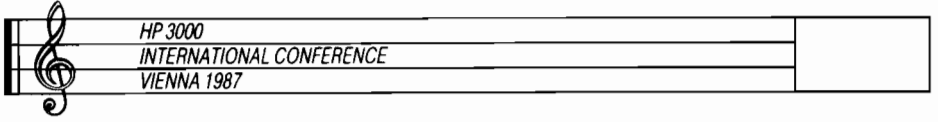
To Bob Barr, Ian Craib, Al Slachta, and Phil Archdeacon, creators of the QUIZ on-line manual.

References


- [Cline et al.] Cline, T.; Fong, W.; Rosenberg, S.: "An Expert Advisor for Photolithography". Procs., IJCAI-85, pp. 411-413, 1985.
- [Fargues and Adam] Fargues, J.; Adam, J.P.: "KALIPSOS: A Text Processor for Knowledge Acquisition". Presented at the IBM Europe Linguistic Seminar, Davos, Switzerland, 1984.
- [Fargues et al.] Fargues, Jean; Landau, Marie-Claude; Dugourd, Anne; Catach, Laurent: "Conceptual Graphs for Semantics and Knowledge Processing". IBM Journal of Research and Development, Vol. 30, No. 1, January 1986.
- [Fikes] Fikes, R.E.: "ODYSSEY: A Knowledge-Based Assistant". Artificial Intelligence, Vol. 16, No. 3, 1981.
- [Fikes and Kehler] Fikes, R.; Kehler, T.: "The Role of Frame-Based Representation in Reasoning". Communications of the ACM, Vol. 28, No. 9, pp. 904-920, 1985.
- [Gomez] Gomez, F.: "A Model of Comprehension of Elementary Scientific Texts". Procs. Theoretical Approaches to Natural Language Understanding, Halifax, NS, 1985.
- [Hiz] Hiz, H. (Editor): "Questions". Dordrecht, 1978.
- [KEE] The KEE™ (Version 2.1) Technical Documentation Set for Xerox (Interlisp-D) machines, IntelliCorp™, Palo Alto, 1985:
 - ° Software Development System Reference Manual, Document Number 2.1-R-7
 - ° Software Development System Users Manual, Document Number 2.1-U1-1
 - ° RuleSystem2 Reference Manual, Document Number 2.1-RR-1
- [Kiefer] Kiefer, F. (Editor): "Questions and Answers". Dordrecht, 1983.
- [Lehnert] Lehnert, W.: "The Process of Question Answering". Hillsdale, NJ, 1978.
- [Matwin et al.] Matwin, S.; Skuce, D.; Szpakowicz, S.: "Question-driven Approach to the Design of a Software Advisor System". Department of Computer Science, University of Ottawa, working paper, 1985.
- [Matwin & Queant] Matwin, S.; Queant, C.: "Knowledge Acquisition by Simple Learning in a QUIZ Programmer's Apprentice". To appear in the Procs. of the 2nd International Conference on Computers and Applications, IEEE Computer Society, Beijing, June 1987.
- [Queant] Queant, C.: "A QUIZ Apprentice". Department of Computer Science, University of Ottawa, Thesis TR-86-13, 1986.



- [QUINTUS] The Quintus Prolog Technical Documentation Set (Version 6), Copyright © 1986, Quintus Computer Systems Inc.:
◦ Quintus Prolog Reference Manual
◦ Quintus Prolog User's Guide
◦ Quintus Prolog System Dependent Features Manual
- [Schank] Schank, R.: "Intelligent Advisory Systems". In: AI Business, P.H. Winston, K.A. Prendergast (editors), Cambridge, 1984.
- [Skuce-83] Skuce, D.: "The LESK Tutorial". TR-83-03, Department of Computer Science, University of Ottawa, 1983.
- [Skuce-86] Skuce, D.: "Natural Language Synthesis of a Database Reporting Language Using Commercial Expert System Technology". Working paper, University of Ottawa, December, 1986.
- [Skuce et al] Skuce, D.; Matwin, S.; Tazovitch, B.; Oppacher, F.; Szpakowicz, S.: "A Logic-Based Knowledge Source System for Natural Language Documents". Data and Knowledge Engineering 1, North Holland Publishing, 1985, pp. 201-231.
- [Sowa] Sowa, J.F.: "Conceptual Structures: Information Processing in Mind and Machine". Addison-Wesley Publishing Co., Reading, MA, 1984.
- [Sowa and Way] Sowa, John F.; Way, Eileen C.: "Implementing a Semantic Interpreter Using Conceptual Graphs". IBM Journal of Research and Development, Vol. 30, No. 1, January 1986.
- [Szpakowicz et al] Szpakowicz, S.; Matwin, S.; Skuce, D.: "QUIZ Advisor: A Consultant for a Fourth Generation Software Package". TR-86-02, Department of Computer Science, University of Ottawa, 1986.
- [Tazovitch] Tazovitch, B.: "Representing Causal Relationships in an Expert Advisor for a Fourth Generation Language". Ph.D. thesis (in progress), Department of Electrical Engineering, University of Ottawa, 1986.
- [Tazovitch-86] Tazovitch, B.: "Representation of Causalities in an Expert Advisor". Working paper, University of Ottawa, December, 1986.
- [Tou et al] Tou, F.N., Williams, M.D.; Fikes, R.E.; Henderson, D.A.; Malone, T.W.: "RABBIT: an Intelligent Database Assistant". Procs. AAAI-84, Pittsburgh, 1984.
- [Wilensky et al] Wilensky, R.; Arens, Y.; Chin, D.: "Talking to UNIX in English: an Overview of UC". Communications of the ACM, Vol. 27, No. 6, pp.574-593, 1984.
- [XEROX] The Interlisp-D Reference Manual, Xerox Corporation, Palo Alto, 1985.
- [Zarri] Zarri, G.P.: "RESEDA: an Information Retrieval System Using Artificial Intelligence and Knowledge-Representation Techniques". Research and Development in Information Retrieval, Sixth Annual International SIGIR Conference, 1983.



HP 3000
INTERNATIONAL CONFERENCE
VIENNA 1987

	HP 3000	A106/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

MACHINE LEARNING
By
David Price BSc
(VRS Software Ltd.)



ABSTRACT.

One of the most salient and most significant aspects of human Intelligence is the ability to learn. However until recently learning has not been noticeable in "Artificial" Intelligent systems. This paper reviews the progress that has been made in this field , some of the problems encountered and possible areas of use, dealing primarily with inductive learning and its application in data analysis. *

MANAGEMENT SUMMARY.

Machine Learning is a specific area in the field of Artificial Intelligence. The techniques developed in this field can be applied to all applications. The main goals of Machine Learning are to develop ways of increasing the efficiency of machine through Learning and to acquire knowledge in a usable way. This paper seeks to outline some of those techniques.

1.0 INTRODUCTION.

Machine Learning is just one of the fields of study within Artificial Intelligence (AI). The study of AI is an attempt to discover ways of making machines appear to perform like human beings, to make science fiction, or at least some aspects of it science fact. Computers have historically done those tasks which are repetitive and time consuming but require no particular type of human skill. During this process large amounts of data have been collected and stored in databases or files. Machine Learning seeks to use this data to learn from. By automating the process it becomes faster and depending on the accuracy of the data and the constraints imposed upon the system something new may be discovered or at the very least rules are formalised in machine readable form, rather than the vague human concepts that we may have.

2.0 AN INTRODUCTION TO MACHINE LEARNING

Artificial Intelligence seeks to give machines a level of intelligence or processing capability, similar to that of humans. The subject has been divided into areas of research which relate to physical and cognitive behaviour.

- a) Knowledge representation.
- b) Problem solving.
- c) Natural and Spoken Language.
- d) Vision and Perception.
- e) Learning.
- f) Robotics.

Many of the above tasks are interdependent and have similar underlying techniques eg. a great deal of the research in the area of machine learning is concerned with the representation of the initial knowledge or facts.

The goal of researchers in machine learning is to construct computers or systems that learn. The first problem therefore is to establish what is meant by learn and then to look at the problems inherent in machine



learning, some of the ways of dealing with these problems and finally the areas in need of continued research.

There are two interpretations of 'learn' that have been used by researchers:

- a) Performance Improvement.
- b) Acquisition of Knowledge.

Associated with these interpretations are five main learning strategies:

- a) Induction.
- b) Instruction.
- c) Analogy.
- d) Deduction.
- e) Rote learning.

One of the most advanced area of study is in the field of Expert Systems or Knowledge Representation where successful products have been produced and are currently in use. Conversely, the field of Machine learning is perhaps the area of least research, while being the most fundamental, after all humans are born being able to do very little, but they learn very quickly to walk, talk, solve problems etc., until after thirty years or so of learning, when some of them are considered to be experts.

Expert Systems rely on rules provided by a human expert which are transformed by a Knowledge Engineer (an Expert System Programmer) into a machine readable form. To gain knowledge (rules) from more than one expert has been shown to make expert systems less 'expert', because discrepancies are introduced which have been resolved by the Knowledge Engineer. Introducing the third person through which all information must be channeled creates inconsistencies. Machine learning seeks to allow the machine to learn from these external sources rather than be told by a Knowledge Engineer. Thus it can use the knowledge that it knows to be accurate and discard or query that which is not, saving time and money.

3.0 LEARNING AS PERFORMANCE IMPROVEMENT.

The key idea in performance improvement is efficiency. That is to say, if a problem solving program was put through an automated learning process its performance would be more efficient.

Efficiency itself is not new to DP professionals, but the means of performance improvement will be. The first approach dealt with here is improvement by moving along the store-v-compute trade off. The second will be improvement by search algorithms.

3.1 STORE-V-COMPUTE TRADE OFF.

This trade off is often domain specific. In some problem solvers the computation of a solution is different for each input. While in others, exactly the same procedure will be performed frequently. In these cases it is worthwhile introducing memory into the problem. Each time a problem is solved an ordered pair of values, representing the problem description and a solution, is stored. This strategy is often called "caching" or sometimes "memo-izing". The benefit of this approach is determined by the frequency of encountering the same problem and the machine cost of executing the problem solving algorithm. Some of the test cases are fairly trivial but the applications in complicated domains should be apparent.

One of the earliest learning systems was Samuel's (1959) checkers learning program. It employed what has now become a standard alphabeta min max search algorithm with a static evaluation function. To accelerate the problem solving Samuel employed the Rote learning technique. This method includes a mechanism for storing board positions as value pairs. Before making a play the program compared the existing and previous situations. Three main learning strategies were used:

1. Playing against itself.
2. Playing against people.
3. Playing book games.

What is most interesting about Samuels work is the way in which he used cache values to compute further cache values. This obviously increased the efficiency of the problem solver. Almost all heuristic search programs now use this last technique.

3.2 SEARCH ALGORITHMS.

Mostows BAR program (1983) is an heuristic learning program that seeks to improve the search for solutions. It uses for its domain the game

of HEARTS. The key point of learning is that portions of the solution test can be "pushed" into the generator of all possible solutions, in order to improve the efficiency of the problem solver. Partial solution tests are stored to be used later. It operates by applying a series of transformation rules to convert the initial problem solver into a more efficient one. It uses means-end analysis to guide the transformation process. The BAR program generates all possible solutions, at each point in the process it maintains a list of all paths under consideration and selects one of these paths to extend and then generates all possible extensions. There are five main steps to the algorithm where information from the solution test can be incorporated into the path-extending heuristic search.

1. The initial sequence can be set to something other than the null sequence.
2. As soon as each active path is created it can be tested.
3. The active paths can be ordered to selection criteria before selecting the next path for extension.
4. Eliminate paths that cannot lead to solutions using an extension test.
5. The extensions can be ordered to some selection criteria before selection.

The final problem solver is more efficient than the initial exhaustive search because only those parts which are likely to lead to a solution are generated.

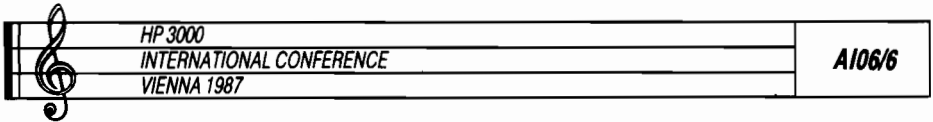
4.0 ACQUIRING KNOWLEDGE AS MEANS OF LEARNING.

The most common ways of giving computers knowledge are through programming, data bases, knowledge bases etc. Another way is to generalize from examples that is to infer rules from facts (training instances). This is usually called inductive learning.

A training instance is a description of an event with an indication whether it is a desired concept. For example, in poker we have

"(5,hearts),(7,hearts),(9,hearts),(jack,hearts),(queen,hearts),
flush."

is a positive training instance.



"(5,hearts),(7,clubs),(9,spades),(jack,hearts,(queen,hearts),
not flush."

is a negative instance.

It is not always necessary to specify a particular desired result as
in this example

(rainfall,sunshine,temperature,pressure,date) where nice
day is (rainfall 0.5) and (Sunshine 6)

where the values of the fields indicate a positive or negative training
example. It is necessary to set up plausability criteria for deciding
which are better rules.

Using these three specifications, the training instances, possible
concepts, and plausibility criteria, the inductive learning system
can employ a generate-and-test algorithm. It is better to use the data
(training instances) to guide the search rather than use a resource
consuming, exhaustive search.

4.1 INFERENCE BASED THEORY OF LEARNING.

Human Knowledge is not static. New Knowledge is added and stored away
and old Knowledge is modified or discarded. We deal with contradictions
by either repairing established knowledge to account for the apparent
contradiction or replacing the knowledge considering it is to be false.
These two approaches can be seen as either revolutionary or evolutionary
and be used when deciding on a strategy for Machine Learning.

4.2 REVOLUTIONARY LEARNING.

Revolutionary learning is fundamentally the disgaarding of a piece of
knowledge when it is shown to be inconsistent and the generation of a
new piece of knowledge. This approach can bring about new and significantly
different knowledge or that which is closely related. That is to say,
two different formulations resulting in similar knowledge. For example
Einstein discovered what he considered to be a new theory but was in
fact a reformulation of Newton's second law of thermodynamics using
different premise.

This theory of learning is also relatively easy to implement on a computer system because no real familiarity with the current body of knowledge is necessary nor is there any requirement for sophisticated knowledge refinement mechanisms. It is important to start from first principles using original facts and observations. EURISKO which learns by discovery is one of the most significant Learning programs which discovered elementary set theory independently from previous mathematical work.

However, this approach has proved to be of general academic interest but often time consuming and inefficient. It is often best suited to limited domains such as mathematical formulae.

4.3 EVOLUTIONARY LEARNING.

Evolutionary learning systems take advantage of knowledge structures which are correct. It would appear to be far more efficient and advantageous to increase existing knowledge by filling in the gaps and modifying that knowledge. This approach can be roughly divided into two main methods, premise-oriented and context-oriented.

4.3.1 PREMISE-ORIENTED METHOD.

This can best be described as "incrementally refining the main body of knowledge". This is the way in which most humans maintain their knowledge base. They make use of what they already know in assessing a new piece of information and revise their opinion accordingly, rather than reformulate all their ideas each time something changes. There seem some obvious reasons for this

- a) The world about us is continually changing and we act accordingly.
- b) Information is given sequentially and we relate it to past knowledge.
- c) Humans do not have a automatic delete function - we may forget but there is no gaurantee. We usually use incorrect as well as correct knowledge.

- d) Memory size determines that we tend to remember the facts that we use most - we rarely forget how to speak or read.

When applying these factors to machines we are not as restricted. It is easy to remove something from a machines memory but perhaps we should not do so. It would be better instead to allow the machine to discard information that is not used in favour of new knowledge. It can also be argued that the availability of vast memories and fast retrieval from strange devices means that there is not such a memory restriction, however it would not be practical to devote large expensive machines to a single application. Therefore good memory management should be applied. This, of course, can bring into question the completeness of the knowledge.

Perhaps the most succesful example of Premise-oriented learning is Michalski's AQ algorithm (the quasi-minimal algorithm). The algorithm generates a set of rules that describes all positive learning events - a 'star' - each of whose elements can be conjunctive concepts (simple concepts). The problem can be formulated as

'Given are a set of rules, a set of newly acquired facts (new learning events), and a set of previous facts from which the rules were induced (old learning events). Then, suppose that some of the learning events contradict the rules. The goal is to transform the original set of rules to a new set, such that the new set is complete and consistant with all the new and old events and should be the most preferred of all such sets'.

As simple a way of looking at this is to break it down into four basic steps

- 1) select an initial set of hypotheses
- 2) generate new hypotheses by generalizing existing ones
- 3) select the best hypotheses generated in step 2
- 4) review step 3 to see if the iteration should continue
(if yes go to 1)

The key point is that the machine decides by some given selection criteria what is the best rules to keep.

Tests completed on this algorithm showed this method to be between 5 and 100 times faster than the non-incremental method. The complexity and the performance of rules learned were on average only slightly worse than those learned in one step. These tests involved the problem domains of chess end games, plant disease diagnosis and insect classification.

4.3.2 CONTEXT-ORIENTED METHOD.

This method is perhaps most relevant for the domains in which we tend to build Expert Systems, in solving complex practical problems for which there are imperfect rules. In this method the cases for when a given rule does not work are gathered together and treated as exceptions, so the production rules

```

if    condition
then  decision

```

can be altered to

```

if    premise
then  decision
unless censor

```

These censored production rules state conditions which, when satisfied, state the reverse of the decision.

Logically equivalent to this is

```

if    condition
then  decision
provided not censor

```

These rules can easily be altered each time a new piece of information (knowledge) is acquired

```

If the ignition key is turned
the the car starts

```

This is discovered, say 20 times, as the driver discovers every work day for a month and is altered to



If the ignition key is turned
then the car starts
unless the petrol tank is empty or the battery is dead
which the system learns one day when the driver is late for work. The
evidence supports this way of declaring the rule because the car
normally starts.

5.0 CONCLUSION.

This paper has looked at some of the theories of Machine Learning but we have omitted discussion in other important areas, including learning of grammars, functions, and procedures from examples and learning from unclassified data(i.e. Clustering). The field of Machine Learning cannot be dealt with in one paper. What has been attempted is an overview of some of the approaches to learning.

In conclusion I will try to outline uses for Machine Learning. These techniques seek to improve on rule-based systems. They can be used to generate rules for Expert Systems or any other rule-based application. The techniques can be used to improve existing applications. The main idea is to solve problems and therefore the domains are not restricted. The most frequent use for Machine Learning has been in the medical field, assisting in diagnosis and research. Machines tend to learn more rapidly than humans (they do not get tired) and can process more information nor do they forget. In most industries there are problems to be solved and automating this process can save time and money.

Machine Learning Systems can be used to generate rules for a rule based system, such as an Expert System. Human Users are often reluctant to state precisely how they make their decisions. The decision making process will often be explained away as experience, a 'gut feeling' or intuition. If the data available to the expert is collected then the computer can learn what the rules for a target expression is. Some experts have exceeded the learning process and therefore do not know how to start explaining their reasoning process, presented with machine generated rules they can assess them and use them as a starting point.

BIBLIOGRAPHY.

- Diettrich, T.G., Machine Learning: Problems and Methods.
Department of Computer Science, Oregon State
University. 1985.
- Forsyth, R.S., Machine Learning Strategies. (in Expert
Systems. ed. R. Forsyth, Chapman Hall 1984.)
- Michalski, R.S., On the Quasi-minimal solution of the General
Covering Problem, Fifth International
Symposium on Information Processing (fcip 69),
Vol A3, Switching Circuits, Bled, Yugoslavia 1969.
- Michalski, R.S., Carbonell, J., Mitchell, T.M., Machine Learning:
An Artificial Intelligence Approach Volumes I & II.
(1983 & 1986).
- Mitchell, T.M., Learning and Problem Solving, Proceedings of
IJCAI 1983.
- Mostow, D.J., Some studies im machine learning using the game of
checkers, IBM. Journal of Research and Development,
No. 3 1959.
- Barr, A., Feigenbaum, E.A. The Handbook of Artificial Intelligence Vol 1.
Cohen, P.R., Feigenbaum, E.A., The Handbook of Artificial Inteligence
Vol 3.

BIOGRAPHICAL NOTES.

David Price is currently a consultant with VRS Consulting LTD.,
involved with system development and support for clients, with a
particular emphasis on AI. He has a degree in Computer Science.



HP 3000

INTERNATIONAL CONFERENCE

VIENNA 1987

THE ANALYST WORKBENCH REVOLUTION

ABSTRACT

The introduction of 4GLs has increased programmer productivity by a factor of 300% to 400%. This has resulted in increased pressure on analysts to produce accurate functional specifications more quickly.

Structured analysis techniques are slowly improving the quality of specification, however the lack of automated tools has prevented their full implementation. Such techniques require an analyst to draw complex diagrams such as data flow diagrams and data models (entity models). Without a workbench the drudgery involved in manually maintaining these diagrams is a headache. In addition a key aspect of structured methods is to prevent inconsistencies and omissions occurring at the various levels of analysis. A good analyst workbench will automatically highlight these inconsistencies, which has previously been the responsibility of the fallible analyst.

The majority of the 20 available analyst workbenches run on IBM AT compatibles, including the HP VECTRA. EXCELERATOR and IEW which at the moment can be considered as two of the big three both run on the VECTRA.

The true potential with the HP environment is to link the analyst workbench to HP's System Dictionary. This will allow information to be controlled on a company wide basis and very importantly provide the input to program development and database generation. Their cost is high, but their arrival heralds a great opportunity for the analyst to throw away his paper and quill.

JIM FARROW and STEPHEN PRICE

DCE INFORMATION CONSULTANCY

Chester House,
Chertsey Road,
Woking, GU21 5BJ.

Prinsengracht 747-751,
1017 JX Amsterdam,
The Netherlands

CONTENTS

- 1) Why the need for Methods?
 - 1.1 To highlight problems quickly
 - 1.2 Provide a means of communication
 - 1.3 Proof of progress

- 2) What are Structured Methods?
 - 2.1 The system development cycle
 - 2.2 Diagrammatic deliverables

- 3) What is an Analyst Workbench and how does it help?
 - 3.1 What should an Analyst Workbench contain?
 - 3.2 What are the benefits?
 - 3.3 What do I need and how much is going to cost?
 - 3.4 The Future for Analyst Workbenches

- 4) Analyst Workbenches and the HP environment
 - 4.1 System Dictionary
 - 4.2 Interfacing the Analyst Workbench to System Dictionary

- 5) Conclusion



1) Why the Need for Methods

1.1 To Highlight Problems Quickly

As the development cycle proceeds from analysis through design to implementation, two phenomena occur. Firstly, the percentage of error in estimation of total project cost decreases. Secondly, the cost of fixing errors increases.

It is therefore vital that the development process is structured to allow critical management, user and technical decisions to be taken at the right time (i.e. early!)

1.2 To Provide a Means of Communication

As the number of people on a project increases, the amount of productive time per day per person decreases dramatically.

Members of the team are spending more time trying to communicate. Checkpoint facilities within a methodology provide a common means of communication.

1.3 Proof of Progress

DP management is under constant pressure to show results, without a methodology all we can do is push for early system completion, thus instead of the project being time shared: 30% Analysis, 20% Design, 40% Implementation and 10% Maintenance.

We save time on the analysis, resulting in: 25% Analysis, Design and Implementation and 75% Maintenance.

But with a methodology we can prove our progress at each step by producing checkpoint documents.

2. What are Structured Methods

A methodology is an integrated set of procedures which provide a complete framework within which a given task can be performed.

A methodology should enable:

- highlight problems quickly
- means of communication
- proof of progress

A methodology should provide:

- guidelines, not rules
- top-down approach
- diagrammatic representation
- standards for use and documentation
- step-by-step procedures

2.1 The System Development Cycle

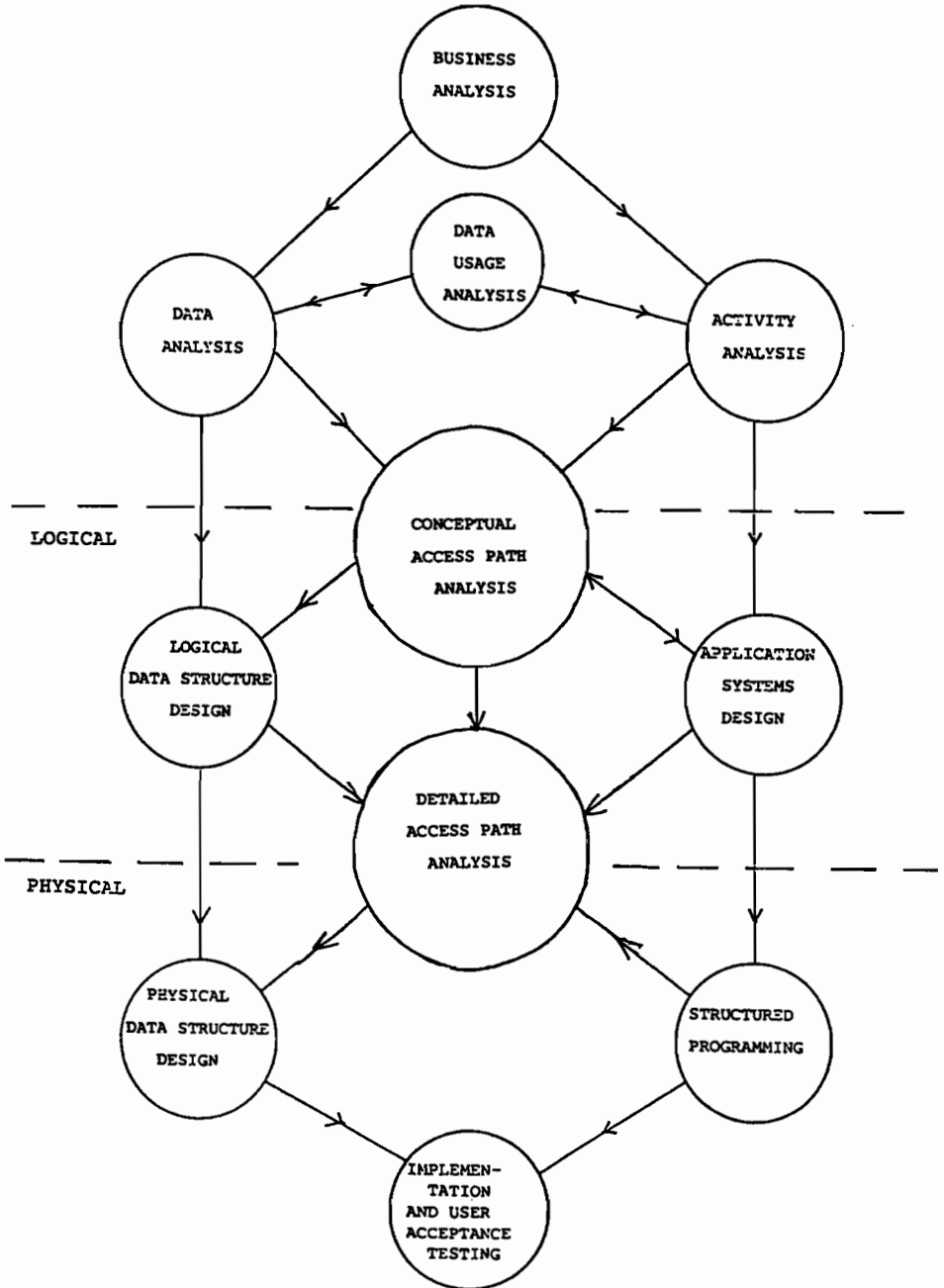
Figure 1 shows how the different parts of a methodology are linked together.

There are three major types of procedure. Firstly Data, which needs to be analysed and modelled. Secondly Activities, again needing to be analysed and modelled. Finally, a method for bringing Data and Activities together.



FIGURE 1

THE SYSTEMS DEVELOPMENT CYCLE (SDC)



Objectives of the Systems Development Cycle Parts

Business Analysis To define the system aims and scope of analysis.

Data Analysis To analyse the data resources.

Activity Analysis To define the user's information handling processes.

Data Usage Analysis To cross-reference and check out the consistency of DA and AA and summarise the access requirements.

Conceptual Access Path Analysis To determine how the activities/processes use the data.

Logical Data Structure Design To map the data model to the logical data structure taking into account any database structuring rules.

Application System Design To translate the user requirements into a technical application system design.

Physical DB Design To translate the logical database into an optimised, workable physical database.

Detailed Access Path Analysis To determine how the processes use the logical database.

2.2 Diagrammatic Deliverables

There are four key diagrammatic deliverables that result from performing analysis.

- i) the data model (or entity model) which records the way in which the business wishes to group items of data such that any person or machine, responsible for performing a business transaction, will have ready access to all necessary information;
- ii) the activity decomposition diagram, a structure diagram showing activities performed within the business, decomposed (broken down) into a number of levels showing greater detail;
- iii) the data flow diagram, which shows the dependency of one activity on another for data. It identifies the sources and recipients of the business data and the cause or triggering event that results in the activity being performed;
- iv) the access path diagram, which shows how the activity navigates the data model and quantifies the retrieval process in terms of number executions per time period.

3) What is an Analyst Workbench and how does it help?

An Analyst Workbench (AWB) is a tool that aims to increase the analysts productivity and accuracy.

3.1 What should an Analyst Workbench contain?

In order to fulfil the above aims an AWB must provide the following facilities: high resolution graphics for modelling; a data dictionary for definition storage and cross-referencing; text processing, including keyword retrieval for documentation generation.

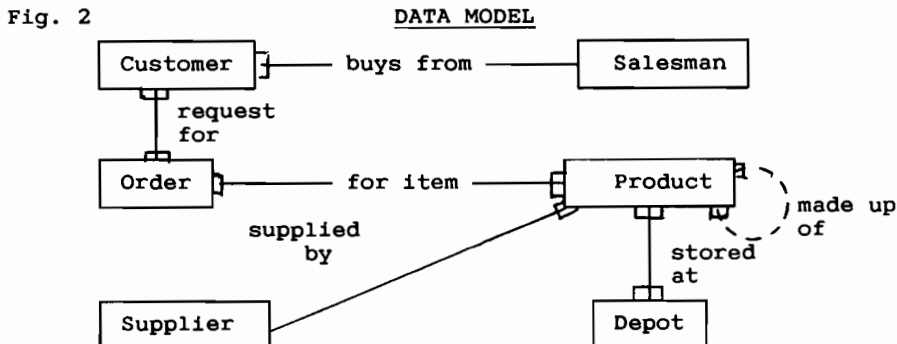
All AWB vary in their levels of functionality and richness. The question is to what degree does it support each function.

DATA MODELLING (Fig 2)

With regard to data modelling a "rich" or powerful data model should allow for:

- * entities including subtypes

- * relationships, displaying the following features:
- many-to-many, one-to-many and one-to-one
 - optional, mandatory
 - involution
 - exclusivity
 - identified by



ACTIVITY DECOMPOSITION

The activity decomposition diagram, besides allowing for activities, should support the recognition of common activities, activity repetition, optionality and the event trigger. (Fig 3) The data flow diagram (DFD) needs to support activities both internal and external to the study scope, plus data flows, data stores, event triggers, sources and sinks. (Fig 4)

At present no AWB supports access path diagrams. (Fig 5)

Fig 3 ACTIVITY DECOMPOSITION

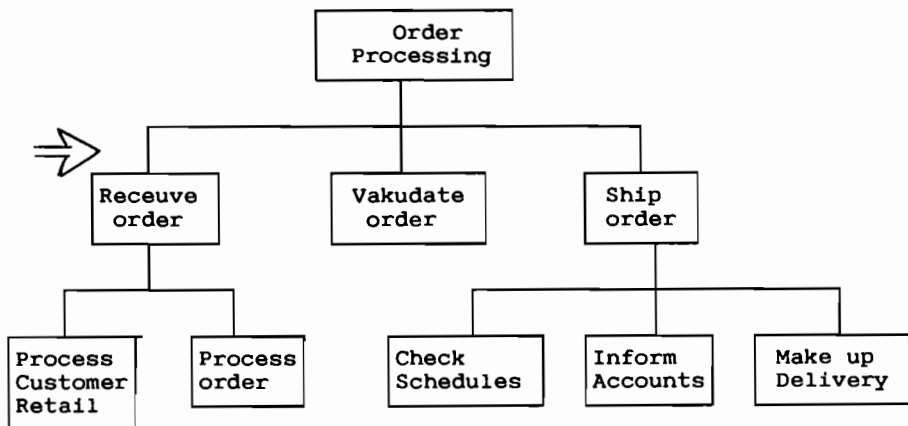


Fig 4

DATA FLOW

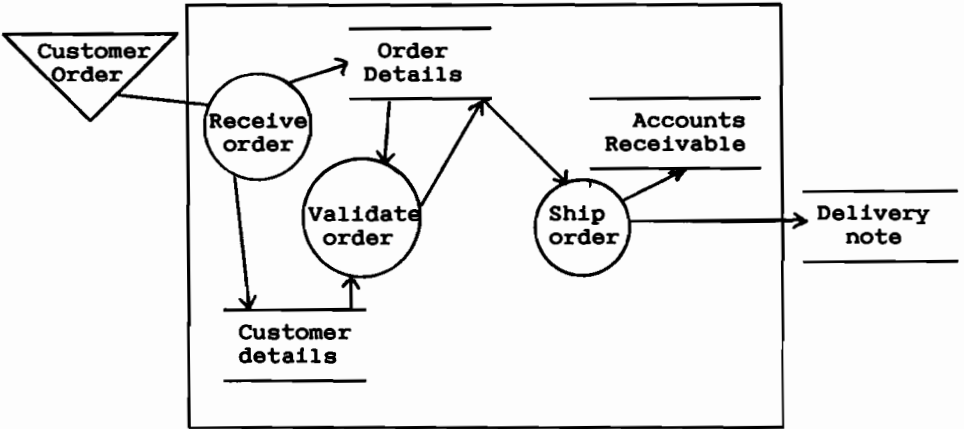
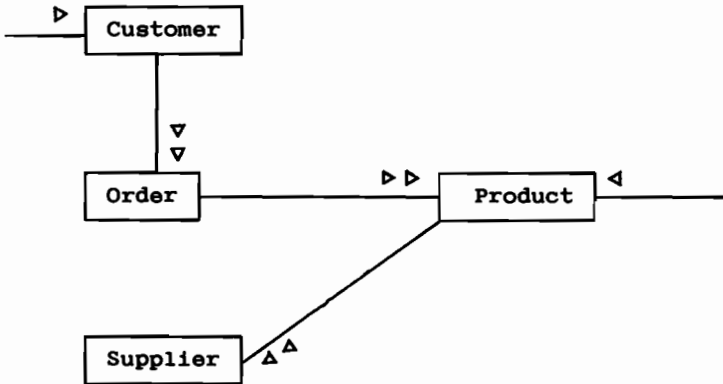


Fig 5

ACCESS PATHS



MANIPULATION

How easy is it to manipulate the diagrams and add detail? For example, can you move from your data model into the data dictionary to describe entities and their attributes? Can the AWB support the explosion or implosion from one level of detail to the next? This applies to all the diagramming techniques, for example, taking a global data model and developing it in detail or exploding a first level DFD into the appropriate number of level two DFDs.

CONSISTENCY CHECKING

How well integrated are the functions. The techniques, if used in conjunction, should cross validate each other, as indicated in Figure 6. In the case of activity decomposition diagrams the activity should be described in the appropriate level of DFD. If an activity is added to a DFD is it automatically added to the decomposition diagram? The DFD uses data stores and data flows which should correlate to entities and attributes recognised by data analysis. If the package does attempt to maintain consistency, what technique is adopted? On-line verification leads to a tighter design, but requires much more processing power and time, whilst after-the-event reporting is less likely to be actioned and water-tight. For consistency checking several products have used PROLOG to set the rules. For flexibility the customer should be able to switch the rule checking facility on or off to suit his particular requirements.

SHARING DATA

Import/Export facilities are desirable, definitions and models from other projects to be downloaded/uploaded as required from a central encyclopedia, but with strict access and update rules.

3.2 What are the Benefits?

The traditional pencil and paper approach is marred firstly by the drudgery involved in manually updating the models. Secondly a manually kept dictionary of definitions relies on the all too fallible skills of the analyst to spot inconsistencies, duplicate names and omissions. Thirdly it is difficult to make the latest version of definitions and models instantly available to other analysts, particularly those based in other offices.

The DCE Information Management Consultancy Group has now used AWBs for about 50 projects, and they clearly do reduce the drudgery of redrawing models and updating definitions.

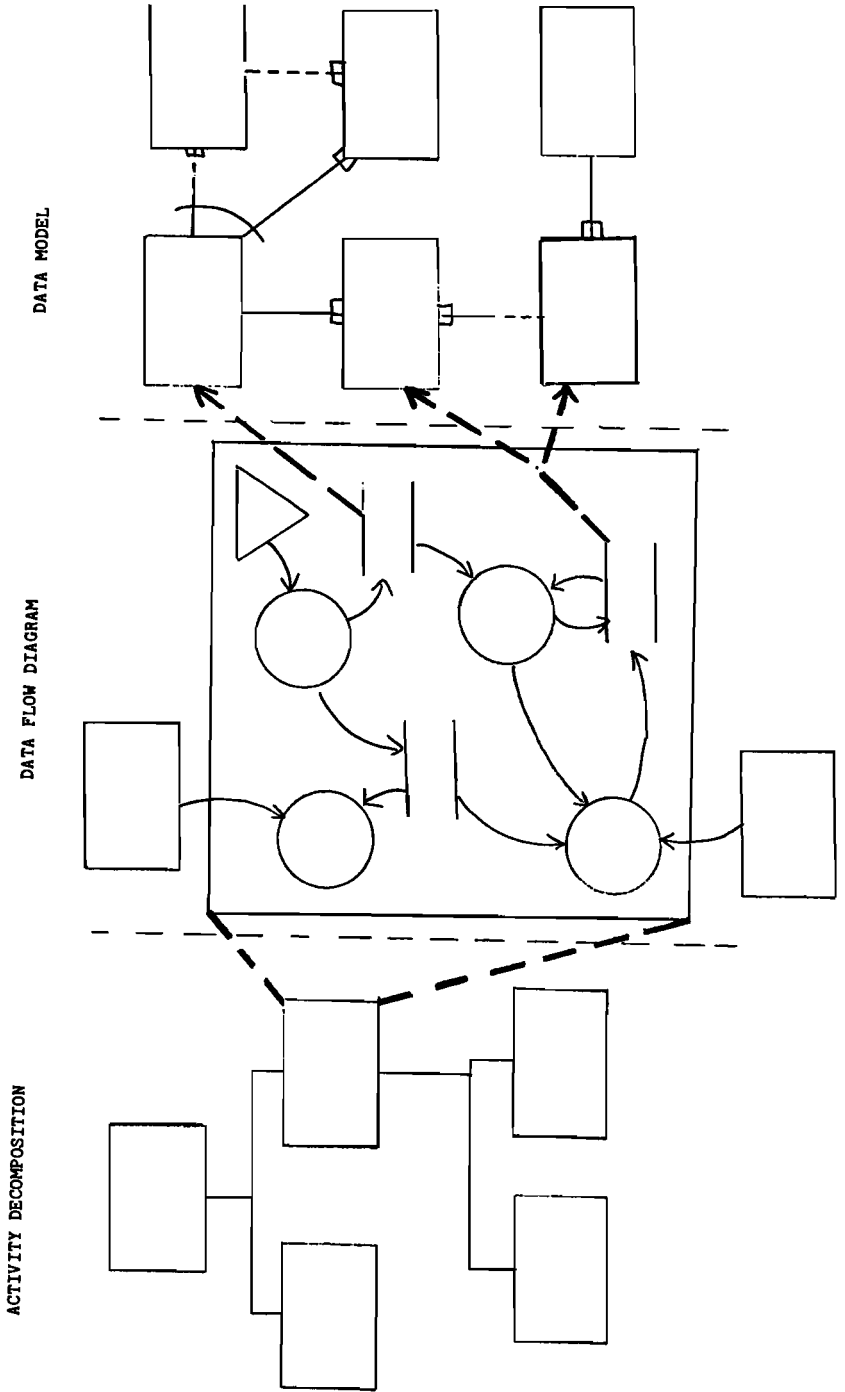
The AWB can be used to introduce a uniformity of standards within a company through use of a common product and training people.

The results of individual projects can then be consolidated to obtain knowledge at the company level. This will take time but will result in better "integrated" systems. Many AWBs can generate standard reports which could be included within a functional specification. Some products, notably Excelerator, are



TECHNIQUE CROSS VALIDATION

FIG 6



close to being able to generate the complete specification to a professional standard. Such a capability is again a step forward as it will aid conformity of standards between projects.

3.3 What do AWB's need and how much will it cost?

Most of the 20 or so AWB run on IBM-PC/AT compatibles (including the HP VECTRA). Excelerator and IEW, which at the moment can be considered two of the big three, both run on the HP VECTRA.

The hardware to support most workbenches is surprisingly powerful and expensive. Typically a PC/AT with 640K to 2M bytes of memory, 20M bytes or more of disc storage, high resolution screen, possibly enhanced colour display, a mouse, plus graphics memory expansion card, and high resolution printer. Representing 5,000 or more for hardware alone.

The cost of the AWB for the micro based products is in a range of between 2,000 and 7,000.

3.4 The Future for Analyst Workbenches

Firstly there will be tighter integration between workbenches and Application Development tools. Initially this will take the form of interfaces, such as currently exist between Excelerator and ICL's ODS dictionary.

Later, the AWB vendors will try to take over the design and construction phases from present Application Development tools by offering prototyping of screens and reports, transaction networks, dialogue/exchange design, and database design.

Conversely, the Application Development Tool designers as represented by, say, Cullinets Ads-Online, will build in front-end workbenches to their own products to squeeze out standalone models.

4. Analyst Workbenches in the HP Environment

WHERE DOES THE AWB FIT IN?

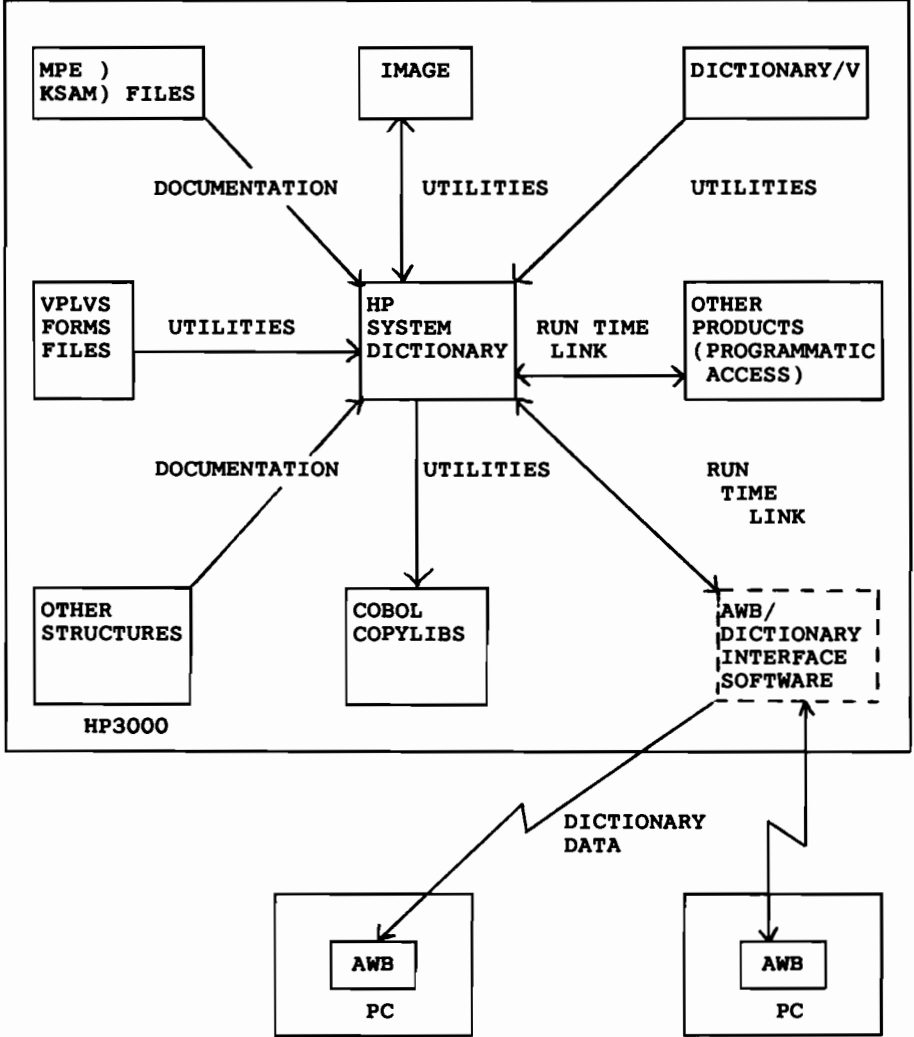


FIGURE 7

In order to reap all the benefits from Analyst Workbenches it is necessary to integrate them into your system. (See Figure 7)

The key to successful integration is the HP System Dictionary.

4.1 System Dictionary

The Hewlett Packard System Dictionary is seen as a key product over the forthcoming years for application development. It is described as being 'extensible'. This means that not only can it be used to record information required for program development but it can also be structured to capture the results of analysis. The first stage involves structuring the dictionary in such a way as to accept the results of data analysis. In essence this requires the recording of entities, relationships between entities and attributes, that define the entity.

Data Dictionary Structure

A standard dictionary structure requires to be built at the outset. The structure, shown in Figure 8, is as follows.

Domains allow for the different business views to be recorded. The COMMON domain will hold the CONCEPTUAL and LOGICAL views. These views are the 'perfect' conceptual world and how that conceptual model will be mapped to a logical DBMS (database management system). This provides the view of where the company wishes to get to. Additional domains are used to describe the current or live situation. This allows a picture to be built showing where the company is, where it wants to be and thus allow for the development of a migration plan.

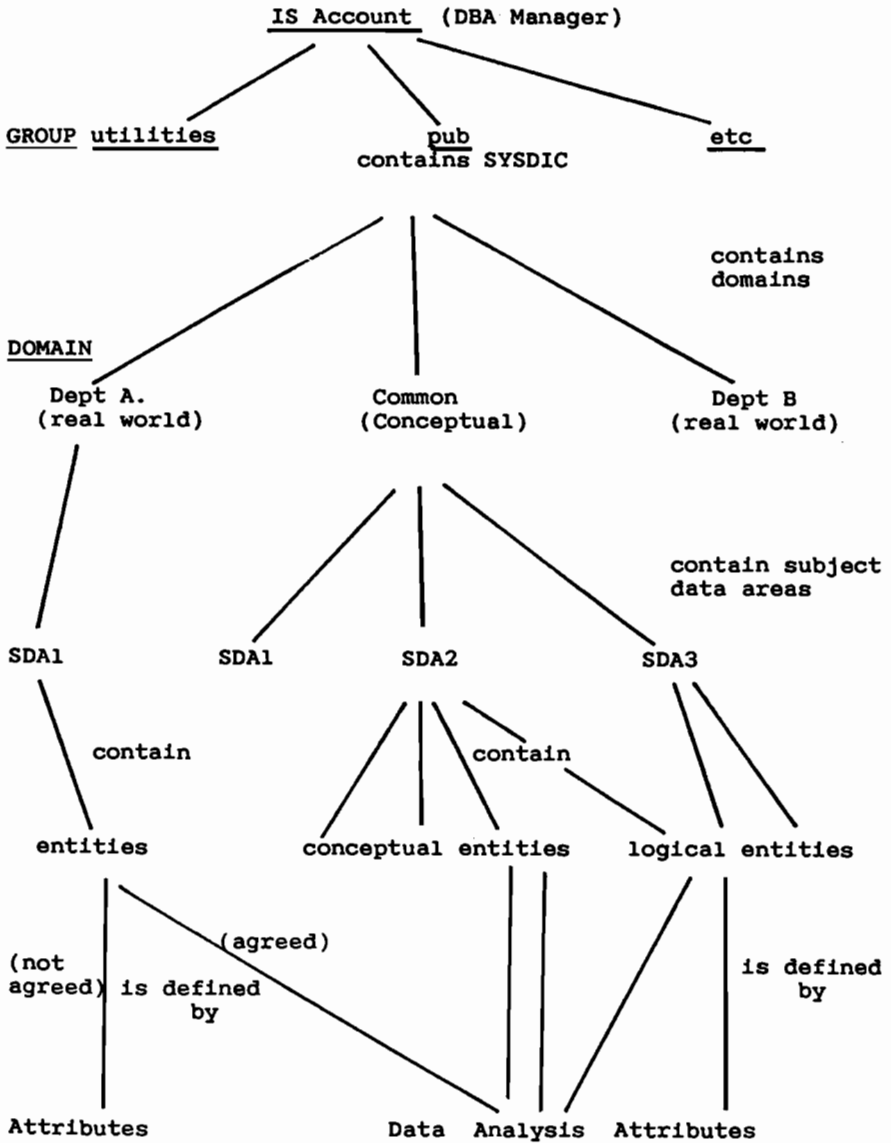
Structure within each Domain

Within each Domain a top down approach will be taken. At the highest level is the Subject Data Area (SDA) which is a high level grouping of entities e.g. FINANCE, CUSTOMER AGREEMENT and THIRD PARTY. The grouping of entities is open to individual interpretation, is Finance and Accounts one or two SDAs, which SDA should the entity VAT appear in?

Initially within the common domain each SDA will contain a number of 'conceptual' entities, as time progresses these 'conceptual' entities will map to 'logical' entities.

Both conceptual and logical entities are described by 'data analysis attributes'. The description of logical and conceptual data analysis attributes are primarily the same. Therefore to prevent having to replicate these descriptions both types of entity are allowed to share the same occurrence of data analysis attributes.

Figure 8 - System Dictionary Structure



4.2 Interfacing the Analyst Workbench to System Dictionary

In the last section we examined a way for allowing HP System Dictionary to capture analysis data. The next step is to have the dictionary populated from the validated dictionary of the Analyst Workbench.

As the structure of the dictionary in the Analyst Workbench will almost certainly be different to that of System Dictionary, it is necessary to have additional software. (See Figure 3).

The interface software would probably be a simple conversion of a subset of the workbench dictionary into System Dictionary format. Although there is no reason why the whole Analyst Workbench could not be catered for, using System Dictionaries extensibility.

Once the System Dictionary has become mature it will feed the new individual projects, who will have the Analyst Workbench and populated dictionary to work with.

5) Conclusion

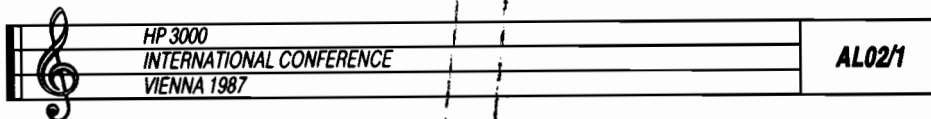
Structured methods allow the analyst to highlight problems quickly, provide a means of communication, and provide for proof of progress.

The Analyst Workbench imposes a standard methodology and gives the analyst productivity gains by removing the drudgery.

The Analyst Workbench also enables the analyst to provide better quality deliverables, by automatic cross validation techniques.

The HP System Dictionary provides an ideal point to link the Analyst Workbench into the HP environment.

Analyst Workbenches are not cheap, but the benefits to be gained now, and in the future, are well worth the outlay. Their effect on the system development life cycle will be more dramatic than that of 4GLs.



System Development and Prototyping Using 4GL's
- The Changing Role of the Programmer -

Jürgen Fritz
JF Management-
und DV-Beratung
Lanzelhohl 34

D-6500 Mainz 1

INTRODUCTION

The Challenge

We may assume that there is no controversy. It is easily to define what the challenge is:

We have to provide the application software that the users need, when they need it.


We will also agree in the review of our real DP world:

Users don't have the systems they need (or want) and what they got came months and years later than wanted.

There feeling is always to be behind the nice ideas they heard about. Ads of all the computer vendors explain them the great chances of computing.

We EDP-Professionals will also agree in the permanently growth of the computer technology. The performance / price relationship is improving from week to week.

When the situation of users in our organizations is not as nice as we want to have it and we can't get it over years, then we might assume that there is something else which must change.

	HP 3000	AL02/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

System Development Productivity

Most EDP-managers understood during the last years that they have to increase the productivity of their development staff. Many of them found that a 4GL is a realistic way to overcome the backlog problem.

But not all of those intelligent managers found after reviewing the year the results which they expected. Some of them thought having chosen the wrong 4GL. May be!?!

Why do we need to improve productivity?

- There is a significant visible and an enormous invisible backlog.
- We can't longer live with the reality of the growth in installation of computer systems and the shortage of experienced, trained professionals.
- Today's development methodologies that of hand in hand with traditional or 3rd generation software tools will just not give us the productivity we need to meet our challenge.

Traditional Methodology

What we followed for a long period of time is a well known concept of development phases which are characterized by:

- Each phase has a certain goal to reach
- One phase is clearly separated from the former and the next one by well defined criteria.
- Different individuals may be responsible for the different phases.

What are those phases?



1. Feasibility Study

- The benefits of the endeavor are evaluated and a project is given the go ahead.

2. Requirements Definition

- Here the functionality of the system and the data structures are clearly and specifically defined. The users review this specification documents.

3. System Design

- The users and the DP professionals meet regularly to redefine the specs and finally, at long last the specs are signed off so programming can begin.

4. Programming and Testing


- Here those programmers are off writing and testing programs to meet the specs and before you can say 'Jack Robinson', voila, the system is done.

5. Implementation

- Now the system is sent out with flags and a brass band blowing to a burst of user applause and an office full of excited and appreciative users.

6. Production

... everyone lives happily every after.

	HP 3000	AL02/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

REAL TRADITIONAL DEVELOPMENT

Now seriously friends!

How many of you find this conversation well known:

"But I'm sure that is what you asked for".

"I know what I asked for but it's not what I wanted".

That is really the key to the inherent limitation of traditional development methodologies.

It sums up why we cannot continue to develop using today's methodologies and tools to meet our challenge.

It is a known fact that depending on the source, 80 to 95 % of the maintenance work in applications is due to errors in requirements definition. Harry Sneed says it with other words: "80 % of the requirement definition is done during the debugging phase". Remember that is the phase of development where users say what they want, the DP professional interprets that and writes down what he thought the user wanted and then the users signed off on what they thought the DP professional meant about what he said about the user wanted.

For every dollar (or shilling) spent to correct an error during requirements definition by meeting again, clarifying, signing off etc. it costs 75 dollars (or shillings) to correct it if it remains undetected until implementation and acceptance testing.

The bottom line is that 60 - 80 % of the maintenance that is done once a system has been installed is done to correct errors that occurred in requirements definition.

What are we doing when we are maintaining the system for these problems?

We are not developing new systems, we are giving the backlog unnecessary opportunity to build up.

Why Errors?

Why do errors occur in defining requirements?

1. First of all it is difficult for users to really visualize exactly what they want.
2. It is often difficult for us to identify all users who will either directly or indirectly be affected by a new system. Those users to often disagree on how things should look.

You see in order to be successful with traditional development we assume that the user knows what he/she wants and is able to describe this in a clear manner.

So the most fundamental aspect of successful system development is Effective Communication.

Assumes

Traditional development methode assumes that users

1. Know what they want
2. Communication is working effectively
3. Requirements are static

Traditional development methode is plagued by

1. Long development times
2. High costs and time for maintenance
3. The build up for backlog

As more computers are installed and the number of trained professionals per installation decreases we need to do something drastically different.

4GL's

They are known as languages which ensure to give results in one-tenth the time of COBOL.

I hope we agree that

"Productivity not a case of simply running fast, but more importantly, ensure that you are heading in the right direction".

Because using a new tool does not automatically mean that the answers will be more correct, it just means you'll make the same mistakes faster (and see them sooner!).

Prototyping

Prototyping is not a new concept. It is an accepted methodology in engineering, manufacturing and construction.

In the text "In Search of Excellence" the authors identify that the first ingredient in a receipt for success is

"rather than have 150 engineers working on a single project which has a planned completion time of 15 months - engineers are split into groups of 5 to 25 people who work closely with clients to develop prototypes in a matter of weeks. The unsuccessful ideas are discarded, the successful ones are implemented".

So then the task at hand is to apply these proven ideas and concepts to the development of application software.

In the "Search of Excellence" example for company success we had the engineer and the customer.

For a software development projects success we have the developer and the user.

So, we begin to build the idea of prototyping by identifying that key players in this new methodology are the user and the developer.

Now, so far it doesn't appear all that different from traditional methodology. If we examine the example a little further we see that it's not just who the key players are that important but how they play with each other.

"working closely ... to develop ... unsuccessful ideas are discarded, successful ones are implemented".

So, successful prototyping depends in close cooperation between the user and the developer.

The user and developer work closely to go from a first functioning version of the application that exhibits all essential features to a final product that meets the needs of the user.

Simple

It all sounds so simple. Why aren't all DP shops that are using 4GL's embracing this methodology called prototyping?

I think it may have something to do with limited liability.

In traditional development the application is defined (the feasibility study), the formal specs are signed off and the programmer codes and tests until the programs meet the specs. At this point if all went well we have success. The specs are the yardstick for measurement and they are static and unchanging.

Hence limited liability if you've met the specs you have done job regardless of the appropriateness of the finished product.

It is often very difficult to adjust to a new mentality. With prototyping an application is continually re-developed and really the job is never finished.

It is a dynamic process of growth and change. There is no longer a fixed unchanging definition of the system to measure your progress against.

How Do We Get

What is it the prototyping really gives us?

The opportunity to reduce the probability of errors in the requirements definition. The prototype itself represents the specifications.

The user sees the system and is now much more capable of deciding whether it is really what was wanted.

How Exactly Prototyping Occurs

Now how does all this happen anyway.

The easiest way to describe the prototyping process is to review the tasks that must be performed.

Users and developers meet to discuss fundamental tasks of the application.

Questions like

- What information do we need to collect?
- What info appears on reports?
- What transactions need to be performed?

"The very basics"

Developer builds from this information the first prototype. It will exhibit all essential features: menus, screens, reports etc. Developer produces documentation (online help!) so that user can systematically evaluate prototype.

User then systematically with document, reviews and evaluates the prototype.

Developer revises the prototype, adds missing essential features and begins firming up database design and logic.

The last three steps are repeated with the key being timeliners. Each repetition will further solidify the database design and the required logic.

With this methodology we are talking about an iterative process that starts as a tool to create a visual representation of users requirements, continues to confirm the feasibility of the application and finally evolves to a state where in fact it is a bona fide production system.

So there we go, the answer to all our problems!

Not so fast!

When Not To Use

There are without a doubt situations where it is wrong to utilize 4th generation tools and the prototyping methodology:

1. When there is lack of commitment

That is, when there is not commitment from management to educate the users in their new, integral role in software development.

2. Also lack of commitment from DP professionals. When they are unwilling or their environment makes them unable to spend the time developing their skills in the 4GL itself and the prototyping concept.

3. When a solution already exists.

Prototyping should not be viewed as a license to re-invent the wheel. It is just a counter productive exercise.

To Consider

If we want to experience the benefits of prototyping an 4GL development tools there are a few things we should consider.


1. Be prepares to deal with the issue of deciding when a system is finished enough to go live.
2. Be prepared to deal with more integral role of users and the greater communication required between users and DP.
3. Be prepared with the changing role of the analyst and the programmer.

Nice, isn't it? Three times I asked you to "be prepared". Easy to say. Also easy to do?

I think to be really successful we have to take into consideration another point.

All these "be prepared" must two sources:

1. Change your mind about your and the users role.
2. Sit down and learn (now together with your friend the user).

	HP 3000	AL02/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	


Programmers typically are technically oriented people. They are working 8 hours per day (and often longer) with the computer. Now the demand comes up to work primarily with people instead of technique. We have to accept that this is a change which might be very hard to understand and to realize for some of us.

There has a process of training to be set up which is more than just learning new keywords and syntax. It is a process of jumping in personally and to give up the one or other prejudice about a methodology I haven't heard about before and more and mainly about the people outside my well known DP world.

The programmer has to understand that his chance to meet the challenge and to increase his personal influence is only realistic if he changes from the programmer (working with the computer on programs) to a consultant (working with users (often with managers) on solutions).

The individual who doesn't see this changing role can't have the success which is normally concerned with the new software (4GL) and the new methodology (prototyping).

But those who take the chance and keep this triangle balanced will see a quite new career path opened and another level of satisfaction during their 8 or more hours job.

	HP 3000	AL03/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Report Generation using a Visual Programming Interface

Tim Dudley
Cognos Incorporated
3755 Riverside Drive
Ottawa, Ontario
K1G 3N3
CANADA
(613) 738-1440


Command language interfaces are not always the most appropriate tool at the initial stages of report design. A loosely constrained graphical notation can be much more useful. Visual programming techniques introduced on the Xerox StarTM, and popularized by the Apple LisaTM and the MacintoshTM, have now made the use of such a graphical notation much more feasible. Also, the direct manipulation techniques described by Schneiderman are now viable because of the wide-spread availability of bit-mapped graphics screens and pointing devices such as the mouse.

This paper briefly discusses visual programming concepts, and then describes the implementation of a visual programming interface (VPI) for a 4GL report writer. The basis for the design is an object-action syntax. A set of icons was designed which represent atomic report entities, and a graphic editor built to manipulate these entities into a report structure. Attribute sheets associated with each of the report entities allow definition of the report entities to the data dictionary. A menu bar controls menus of all possible actions to be performed on the objects. A facility to switch easily between the graphical and textual representation of the report is provided, with direct manipulation editing available in both representations. Modifications made in one representation are automatically reflected in the other.

The combination of the VPI with a 4GL makes the design and modification of reports remarkably straightforward, and suitable both for end users and application programmers.

Introduction:

In late 1984 and early 1985, Cognos Inc. was involved in a consulting contract with the Ice Branch of Environment Canada, to produce a conceptual design, functional specification, hardware-software specification, and detailed implementation plan for a system which was to archive all available information on sea and lake ice, and iceberg conditions, in Canadian and adjacent waters [Dudley 86]. Two constraints had a considerable impact on the approach that was taken in the design of the system: only two people were available to produce the work, and the work was to be done over an elapsed time of four months, including the Christmas and New Years holidays. The limited resources and timeframe forced us to rely primarily on computer-generated diagrams, made up from a minimum closed set of icons (which we designed), as the basis for the work. We simply didn't have time to produce a textual specification. We found that our design approach was completely altered as a result. In the process of cleaning up system diagrams, we discovered connections in the diagram that were incorrect. Some areas of the diagram had become extremely cluttered; attention was being drawn to those areas, strictly because of their visual appearance. By rearranging the diagrams to

	HP 3000	AL03/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

eliminate the clutter, we were able to remove unwanted redundancies, minimize the number of interconnections between entities, and in effect produce a "canonical form" drawing of the system. One system diagram was shown to a colleague who had been involved earlier in the project. He looked at the diagram for about fifteen seconds, and asked where the connection was between two of the system modules, knowing that that connection had been part of the original User Requirements. He had found a mistake in the design, which had appeared visually as a blank area in the system diagram. At that point, we realized that we could literally design graphically at the high level. We also realized that we could concisely communicate a tremendous amount of conceptual information, by primarily using diagrams illustrated by text, rather than by using text illustrated by diagrams. The diagramming also forced us to modularize our design, and do it fairly rigidly, while at the same time without overspecifying or overconstraining the individual modules or their interfaces, which could have imposed hidden restrictions on the design.

The Hardware/Software Alternatives portion of the project posed another problem for us which we were able to resolve graphically. We had determined that the system hardware block structure consisted of five modules. We needed to ensure that these five modules supported the complete functionality of the system we had designed. We were able to graphically superimpose a system block structure diagram over the functional specification diagram. This was done visually, by working with the shapes in the functional diagram, and mapping the five structural blocks onto those shapes. When we were able to successfully come up with a clean mapping of the block structure diagram over the functional diagram, and were able to draw it, we became increasingly confident that our design was sound.

Our experience with using icons to design this system has prompted us to investigate the potential of applying a similar approach to business applications. Graphics hardware, particularly bit-mapped screens and pointing devices such as the mouse, have now become widespread and relatively inexpensive. Windowing systems are becoming common, and object-oriented programming is fairly well understood. This combination of events has resulted in some interesting software development techniques, particularly in the areas of direct manipulation and visual programming. The remainder of this paper describes our current research, utilizing these techniques, toward the development of a visual programming interface (VPI) to a report writer.

Background:

The terms "visual programming" and "program visualization" are sometimes used to refer to the same thing, when in fact they represent entirely different concepts [Meyers 86]. According to Meyers, visual programming refers to a system that allows a user to specify a program graphically, while program visualization, on the other hand, allows a conventional, textually-specified program to be viewed graphically. This distinction is blurred in the literature, but can be easily remembered by thinking of visual programming as the specification stage of programming, and of program visualization as the documentation or analysis stage.

Another important concept in this context is that of "direct manipulation" [Schneiderman 83]. Direct manipulation is the set of principles which include visual representation of the objects of interest, selection and physical actions instead of keyword commands, and rapid incremental reversible operations [Schneiderman 86]. It is the principle used by such systems as the Xerox Star™, and the Apple Macintosh™ and Lisa™, as well as most video games. It lends itself well to object-oriented programming, and is sometimes referred to as the "point-and-shoot" approach. This is the approach in which the designer selects an object (points), then causes some action to be performed on the object (shoots). An example of this point-and-shoot technique in a word processing application is to highlight a block of text, and then to choose a CUT or COPY or DELETE action from a menu. This

approach can be very straightforward and easy to learn. File manipulation becomes almost automatic: To delete a file, one drags a picture of it onto a picture of a trash can. No more trying to figure out whether the command to delete a file is **DELETE**, **DEL**, **REM**, **X**, **KILL**, **RMFILE**, and so forth. The principle at work here is that our reading vocabularies are considerably larger than our speaking or writing vocabularies, and that we can do more error-free work by pointing at things and moving them around, than we can by writing about them.

The visual programming and direct manipulation techniques also make it easier for a designer to present a system to the user in the user's own framework, rather than in computer-ese. Users "...develop conceptual models - mental representations of the workings of the system." [Rubinstein 84]. The user's conceptual model must correctly predict the behavior of the system. The system designer must therefore anticipate the conceptual model, and present a consistent *external myth* which will reinforce it. (The reason the word "myth" is used is because it is a representation of the internal workings of the system, and may not correspond to the actual internals of the system. [Rubinstein 84]). If the user's conceptual model corresponds well to the designer's external myth, the user will be able to deal with his problem at a higher level of abstraction, and not get mired in the workings of the application or the user interface. The use of icons and their direct manipulation lends itself well to the presentation of an external myth. The most common example of an external myth is the familiar desktop metaphor.

Design of the VPI:

The visual programming interface which we are designing is based on a noun-verb-adjective/adverb syntax, in which nouns are represented by icons, verbs by menu items, and adjectives and adverbs by property sheets and dialog boxes. This syntax is presented to the user, using the desktop metaphor, as shown in Figure 1:

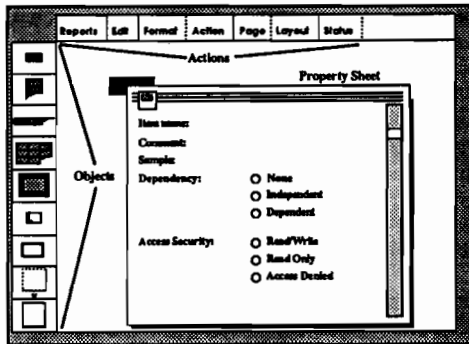


Figure 1

The user creates a report specification by selecting the appropriate icons from the icon menu, arranging them on the desktop workspace according to how the final report is to look, and defining each of the report elements (represented by the icon and its attributes or properties) to the dictionary, through the use of menu selections and property sheets.

Each icon has an associated property sheet, which can be considered as the window into the dictionary for the particular report element represented by that icon. The report writer itself is completely defined by the determination of what report elements are made available to the user through the icon menu, which actions are available through the menu bar, and what attributes are available on the property sheets associated with each report element. Dialog boxes are available for

actions which require clarification. The combination of icon, menu items and property sheets must present a consistent graphics vocabulary to the user in order to be effective. The icons used in our VPI, and their definitions, are listed in Figure 2:

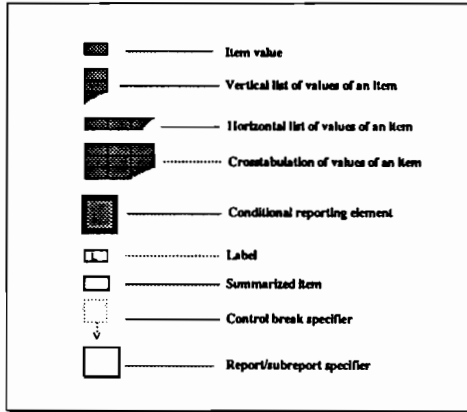


Figure 2

(The design of the icons, and in fact the design of the whole desktop, is a critical part of the success of a visual programming interface. This issue is addressed in some detail in [Verplank 86]. The icons must visually resemble the report elements which they represent, and give the illusion of directly manipulable objects. The whole desktop needs to present visual order, and provide user focus. In addition, the entire system needs to reveal a structure which is consistent with the user's conceptual model, so that the user always knows where s/he is, and what will happen if s/he hits the DELETE key [Goldberg 86], [Verplank 86].)

The best way of describing the VPI is with an example. (For this example, it is assumed that the hardware includes a bitmapped graphics screen and a single-button mouse.) Suppose a report is to be created which consists of a sorted list of an organization's employees and their telephone extension numbers. The telephone numbers are four digits long, the first digit of which indicates the floor on which the employee works. The list is to be sorted alphabetically, by floor, with appropriate titles.

Using the VPI to produce the structure for this report, the user creates the diagram shown in Figure 3:

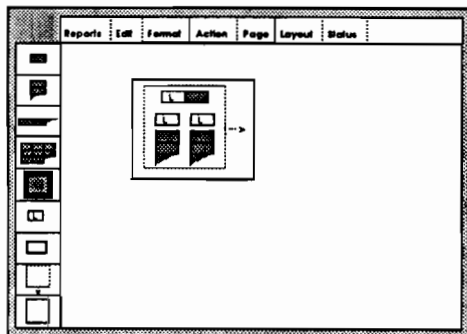
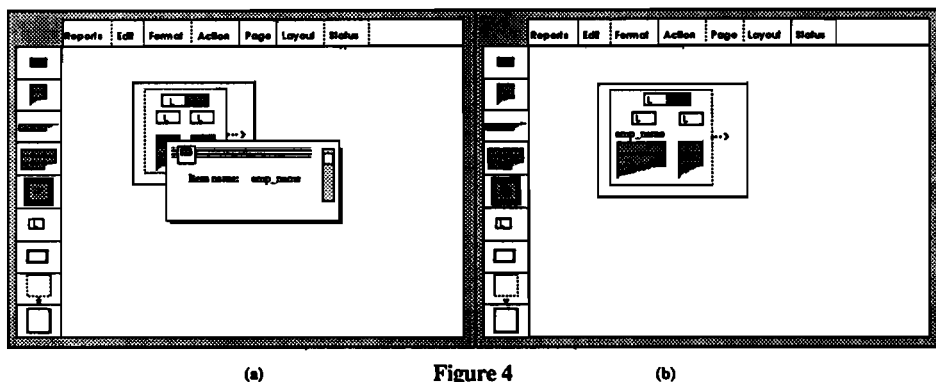


Figure 3

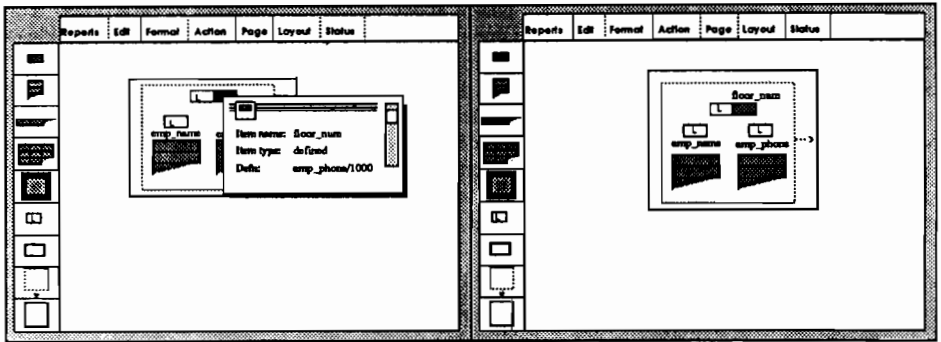
Icons are placed on the desktop by clicking on them in the icon menu with the mouse, then dragging them into place on the workspace. In this example, vertical list icons are placed next to each other, representing lists of the employee names and the employee extensions. Label icons are then placed above each list. A label and item icon are placed above that group, adjacent to each other. This picture represents the portion of the report for one floor. Because this group is to be repeated for each floor, the control-break-specifier icon is placed around it. (In the cases of the control-break-specifier icon and the report/subreport-specifier icon, once they are placed on the desktop workspace, they can each be selected and stretched to any rectangular shape, in order to enclose other icons.) The direction indicator on the control-break-specifier icon is set to point to the right, indicating that the group is to be repeated horizontally, instead of vertically. The report/subreport-specifier icon is then placed such that it encloses the entire group. The resulting diagram defines the structure of the report, and all that remains is to identify each of the report elements to the dictionary.

Figure 4 shows how items are identified:



The property sheet for the left list is brought up by double-clicking the icon (Figure 4a). When the property sheet is displayed, the user keys in the item name for the list. If that item is defined in the dictionary, its attributes are placed on the property sheet. The user then clicks the close box in the upper left corner of the property sheet. The resulting picture (Figure 4b) indicates that the item is defined by displaying the item name at the top of the list. The icon is also expanded by the system to the size necessary to correspond to its size attribute in the dictionary.

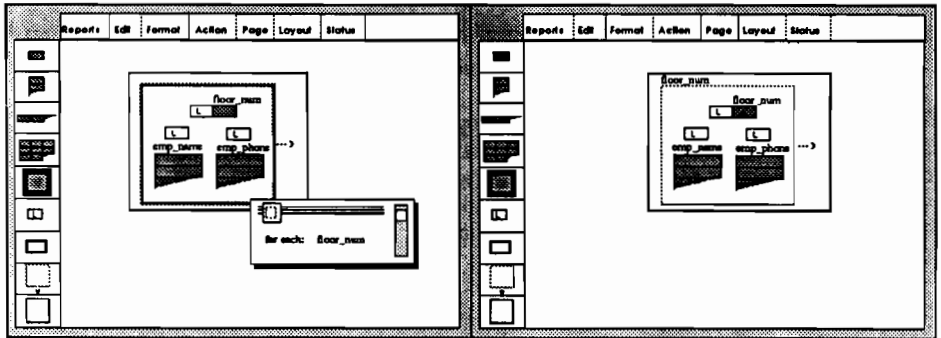
Figure 5 shows how items are defined which aren't already in the dictionary:



(a) Figure 5 (b)

The item icon in the title for the repeating group is double-clicked, bringing up its property sheet. When the item name is keyed in, and the name is not found in the dictionary, the system prompts for a definition. In this case, the item is the floor number, which is calculated as indicated on the property sheet in the diagram. Default attributes are assigned, depending on what is keyed in. The resulting picture is shown in Figure 5b.

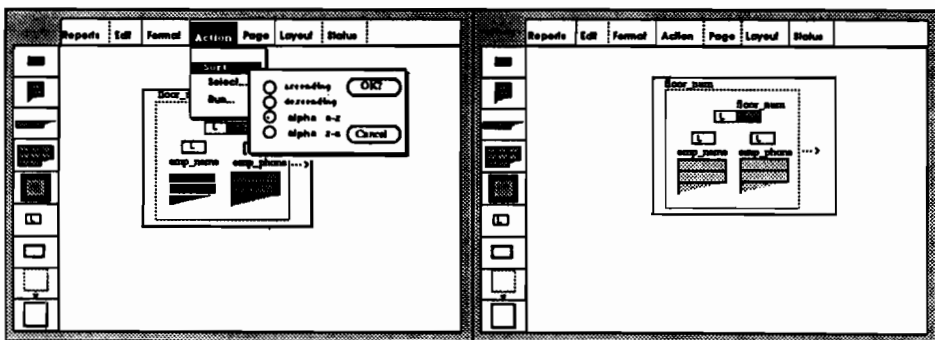
Figure 6 shows how control breaks are specified:



(a) Figure 6 (b)

The control-break-specifier icon is double-clicked, bringing up its property sheet, and the control break variable is keyed in (Figure 6a). Note that the control break variable does not have to appear on the report itself. The resulting picture is shown in Figure 6b.

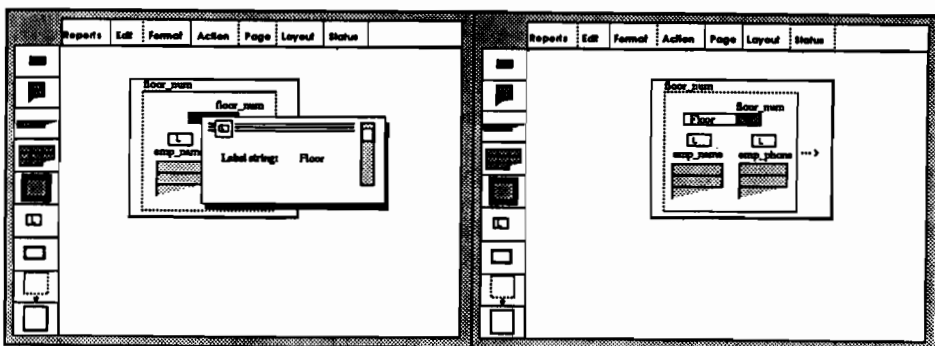
The employee list is to be sorted. This is an action, which is invoked as shown in Figure 7:



(a) Figure 7 (b)

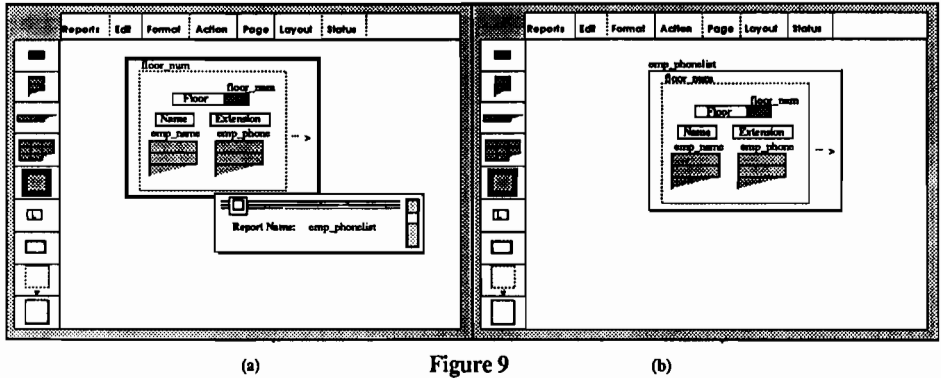
The employee name list icon is highlighted by clicking on the top item, then dragging the cursor down the list. This selects the entire list, as opposed to just the first item in the list. While the list icon is highlighted, the Action menu is pulled down, and the Sort item is selected (Figure 7a). This causes a dialog box to be displayed, asking what type of sort is to be performed. Because the list associated with the selected icon is alphabetic, the numeric sort options are disabled. (This is an example of how the system can be constructed to prevent the user from making errors.) When the OK? box is clicked, the menu is hidden, and the resulting picture is displayed (Figure 7b). Note that the fill pattern of both lists has been changed. This shows two things: that some action has taken place on the indicated report element, and that the indicated report element has some dependencies on other report elements. The actual dependencies are not shown on the report structure diagram, but are available on the relevant property sheets.

Figure 8 shows how label strings can be defined. Note that the string value appears inside the icon, becoming part of that icon. The icon is also expanded to accommodate the string. The font and size of the string can either be set or modified on the icon itself, by highlighting the icon, then selecting the size and font, or on the property sheet.



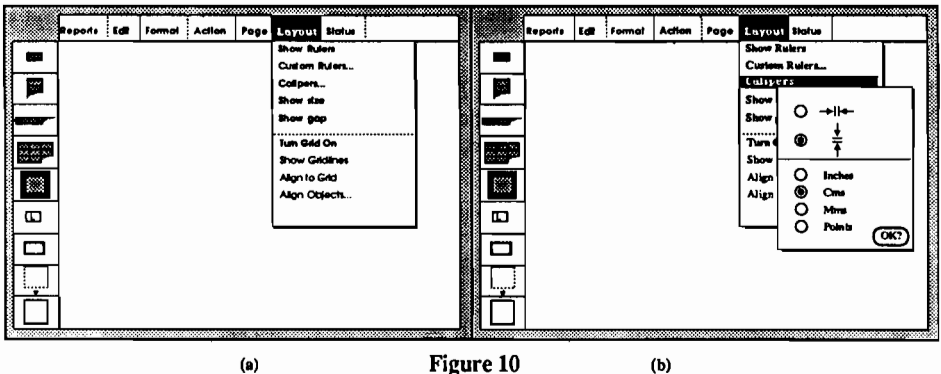
(a) Figure 8 (b)

Figure 9 shows how the report/subreport is named:



Once completed, the report/subreport can be saved by use of the Save item in the File menu.

All physical layout on the page in the above example is done by defaults in the system, depending on the relative spacing of the icons in the report structure diagram. However, in the case of pre-printed forms, certain layouts are predetermined, and the report must fit the layout. The Layout menu provides the facility for accurately placing report elements on a page, through the facilities of grids, rulers, and calipers (Figure 10).



Rulers can be displayed across the top and down the left side of the screen, and can be set to a variety of units (centimeters, inches, points, etc.). As the cursor is moved across the work area, the current cursor position is tracked on both of the rulers. Used in conjunction with an enabled grid, this makes it quite straightforward to accurately position report elements on a physical page.

The calipers provide a mechanism for directly specifying distances between report elements, and sizes of report element fields. (The caliper icon does not appear in the icon menu, because its use is one of action, and it is not part of the report structure.) The calipers are used by selecting the horizontal or vertical caliper icon from the Layout/Calipers menu and dragging it onto the work area. One end of the caliper can be locked by clicking on it, and the other end positioned by dragging to the desired (horizontal or vertical) position, then clicking on that end to lock it. If the Show size or Show gap menu items have been selected, the distance spanned by the caliper will be continuously displayed between the caliper ends while the caliper is being set. If not, it will not be displayed until both ends are locked into position. In order to force a dimension onto the caliper, the user locks it into position, then clicks on the displayed dimension, and keys in the desired dimension. The caliper will be adjusted to the new dimension by the system, and can then be repositioned as desired.

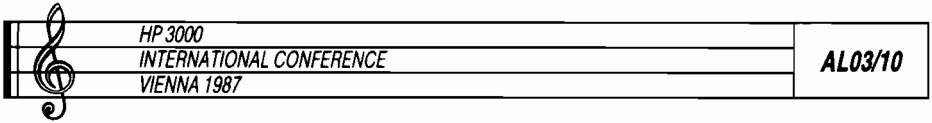
The actual positions of report elements can be seen on the property sheet associated with the report/subreport icon, or by highlighting an icon or pair of icons and selecting the Show size or Show gap menu items from the Layout menu.

Some Problems:

One of the major principles in the design of user interfaces is "Know thy user". This presents some severe difficulties in designing an interface to something like a report writer (which is in fact a graphical language), because of the diversity of potential users. These users range from the hacker/guru, to the application programmer, to the Vice President of Finance, to the CEO's administrative assistant. Each of these users approaches the system with a different conceptual model, and with different expectations of how to use it. In spite of the popularity of systems that utilize direct manipulation and icons, many of these users simply don't take them seriously...they don't believe that such systems provide enough flexibility to allow them to do what they want to do. This, in turn, poses the question of whether or not to design a closed system that does only the tasks which have been specified for it in the Task Analysis (which is another of the major principles of system design...).

We are addressing this problem of appealing to a diversity of users by providing several interfaces, and making it easy to switch among them. The Visual Programming Interface produces an internal report definition, which can then be edited using a syntax editor, or which can be run to create a report. The report itself can then be manipulated using the What-You-See-Is-What-You-Get (WYSIWYG) principle, in combination with direct manipulation techniques. If the report structure is not satisfactory (in the example above, maybe the control break group should be printed vertically rather than horizontally), the system provides the capability for the user to switch to the interface which is best suited for making the required changes. (Incidentally, this change is extremely easy to make using the VPI...one just changes the direction of the control-break-specifier icon. The system does the remainder of the formatting.) The problem that this approach (the provision of several interfaces) leads to is how to make a smooth transition between interfaces, and how to maintain an internal representation of the report which can be efficiently operated on by all three interfaces.

Another problem with which we have been dealing is that of conditional reporting. One of the underlying principles of the VPI is that people recognize documents initially by their visual appearance...that is, how they are laid out. The VPI takes the approach that the user lays out the report how s/he wants it to look, then goes about defining each report element to the dictionary. However, the case arises in which the report format may change, depending on some condition. In other words, there may be instances when a report has three columns in a group, but other instances where it may only have two. Our first attempts to solve this were seriously frustrating, and we eventually decided that it couldn't be done in the VPI context. We have subsequently decided to



include the concept of a Black Box, (and developed an icon for it), to represent a conditional reporting situation. Our current thinking is that the Black Box will appear in the report structure wherever there is a conditional reporting situation. When the Black Box icon is opened (by double-clicking, for example), the alternative report structures will be displayed, as will the determining condition associated with each of them. At the time of this writing, this problem had not been fully addressed.

The conditional reporting problem causes some philosophical consternation with the VPI. The diagrams generated using the VPI were intended to be analogous to the schematics for an electrical diagram or printed circuit board. The idea was that the diagram was essentially a software schematic...a diagrammatic representation of the actual report, which was recognizable immediately, and fairly clearly understood, because of its visual shape (the idea of "revealed structure" again). However, the analogy suffers with conditional reporting, particularly when the conditional reporting variable doesn't physically appear on the report. The analogy suffers further, because a printed circuit board schematic doesn't necessarily look like the finished board, but we are saying that the report structure "schematic" strongly resembles the finished report. We are still struggling with this one.

Future Work:

At the time of this writing, the implementation of the actual VPI was just beginning, and consequently, we have not yet been able to test our ideas in a prototypical environment. We believe that we will benefit considerably from building the prototype, and will be able to refine the design to provide an excellent interface. The main area where we expect to learn is in the definitions of the actions which appear in the menu bar. Because we are building a closed system with this interface, we must ensure that this set of actions is at least necessary and sufficient, and we can't expect to determine a priori whether this is the case.

We have not yet decided how to handle file access and linkages. We are considering having the system infer which files are required, and how they should be linked, from the report structure. Other alternatives are to have the user select file names from a scrolling dialog box, or to provide another desktop at the file level, and have the user specify the required files and linkages graphically.

We also haven't finalized the characteristics of the property sheets...what goes on them, how they are organized, and how they interface to the dictionary. There is also the question of whether the property sheets should look the same to all users, or if they should be different, depending on the user's security access to the dictionary.

When the interface is stabilized, we expect to have learned enough about how to deal with visual programming and direct manipulation to provide a VPI for the entire STORM environment, and potentially to operating systems in general.

Summary:

We have designed, and are in the process of building, a visual programming interface to a report writer. This interface will allow users to graphically design report structures, which can then be executed to produce finished reports. We believe that this approach makes the design and modification of reports remarkably straightforward, and can significantly improve the productivity both of application programmers and of end users.

References:


- [Dudley 86] Tim Dudley. "Graphics in Software Design." *Computer Graphics World*, 9(2), February 1986.
- [Meyers 86] Brad Meyers. "Visual Programming, Programming by Example, and Program Visualization: A Taxonomy." *Human Factors in Computing Systems: Proceedings SIGCHI '86*. Boston, MA, USA. April 13-17, 1986
- [Goldberg 86] Adele Goldberg. Keynote address, *SIGCHI '86*. Boston, MA, USA. April 13-17, 1986
- [Rubinstein 84] Richard Rubinstein and Harry Hersh. *The Human Factor: Designing Computer Systems for People*, Digital Press, 1984
- [Schneiderman 83] Ben Schneiderman. "Direct Manipulation: A Step Beyond Programming Languages," *IEEE Computer*. 16(8) August 1983*
- [Schneiderman 86] Ben Schneiderman. "Direct Manipulation: An Object-Oriented Visual Programming Language," *Human Factors in Computing Systems: Tutorial 17. SIGCHI '86*. Boston, MA, USA. April 13-17, 1986
- [Verplank 86] Bill Verplank. "Designing Graphical User Interfaces", *Human Factors in Computing Systems: Tutorial 1. SIGCHI '86*. Boston, MA, USA. April 13-17, 1986



HP 3000

INTERNATIONAL CONFERENCE

VIENNA 1987

	HP 3000	AL04/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The Human Interface to 4GL.

Abstract

This paper describes the change in application development due to the availability of 4GL. It describes the position of the end-user as well as that of the DP professional and explores the critical success factors and hazards. It includes experience with ARTESSA/3000.

1. Definitions

"Human" in the context of my address, means individuals who are able to communicate in such a way that they understand each other i.e. they speak the same language, and have a strong desire to understand each other. The word "human" presented at a computer show also means that we as DP professionals are becoming more and more interested in human beings, and the way they communicate with their systems. We are coming down from the "Ivory Tower" of data-processing.

Finally we are coming to a situation where users decide whether they want to use a computer, how and when, and for what purpose.

Today, the computer is de-mystified due to widespread home computers, PC's on workers' desks, and better education in this field.

Market research in 1975 showed that every family in the US used 20 to 30 electric motors in their household, varying from hairdryers to lawnmowers. Nowadays, the average population of silicon chips per household is already 3 to 5 and expected to grow dramatically. This growing consumption is due to the improved user-friendliness of these devices. We soon shall hardly recognise that a chip is incorporated in a device, as they behave in a more "human" manner.

How this should happen can be described under the definition of the INTERFACE. This needs a description of how to communicate, by what means, and what the subjects are. An understanding of the process is required, in order to obtain benefits from the interface.

What is an example that more or less describes the function of an interface? When a long distance call is in progress, the interface shall be a common language. In a situation where there is face-to-face contact, for example, you need a slice of bread, gesture or "body-language" will often suffice to explain. Again, a common language.

We make the assumption that all participants are willing to listen to each other, and to understand.

This leaves the question: what is a 4GL? Whatever we may think today, the judgement rests with the historians of the future. After the machine-language and assembler era, and pending the era of procedural languages, we were able to identify the first and second generation. By procedural languages I mean COBOL, FORTRAN, Basic, Algol, etc. Commonly we have decided that these languages are of the third generation.

The need for classification became urgent when the so-called fourth generation appeared on the horizon. In what way is a 4GL different from a procedural language ? Perhaps the following statements clarify the definition:

- * **Integrated and Compatible.**
4th Generation Languages must be able to integrate easily with other tools, so that the whole appears as a coherent set of tools, for the (end-)users. Technically speaking, there must be common components, and compatibility of functions and language.
- * **Powerful Man-Machine Dialogue.**
The state-of-the-art allows us to instruct the computer using non-procedural languages, but we should not have to cope with a glossary of unfamiliar and incomprehensible terms. The better system helps you to check your input.
- * **Transparent and Portable.**
The physical storage of data in a computer is of no concern to a user, so the user should not be burdened with knowledge about the underlying data structure. The 4GL must even protect the user from the operating system and its incomprehensibilities. The 4GL should be completely problem-oriented and environment independent. So the installation of a new operating system, or even new hardware, should not hamper the functionality of a system.
- * **User Friendly and designed to cope with Human Factors.**
The software must forgive user errors. It must organize information in the same way as a user sees it. "Key paths" are of no concern to a user, he just simply wants to retrieve information from the system.
- * **Central and Generalized Data Dictionary.**
There must be a Data Dictionary which is flexible, and able to store all information, with respect to status and relationship of processes, screens, reports and users. It must be generalized enough to integrate with other data dictionary systems and, for example, screen creation programs and report writers.
- * **Making the work of the DBA more Efficient, Reliable and Easier.**
The 4GL should include a comprehensive set of utilities to control and monitor database administration. In every DP shop of a certain magnitude, a tool like this is necessary for proper operation.
- * **Screen Creation.**
A 4GL should provide facilities for easy screen mappings, independent of the programs that use them. It should provide possibilities for the easy layout of screens, and setting attributes to the screen such as high-lighting, security, validation and editing. The data dictionary should indicate the programs using those screens.

* Display Screen Management.

The software should treat a terminal as a set of one or more logical screens. This capability is important when several programs are to display data independently to a user.

* Easy-to-use Editor.

This is an essential facility to ensure that only syntactically correct statements are entered. It must provide insert, delete and update modes. A "fill-in-the-blanks" interface generally is the most effective.

This brings me to the next chapter describing the various types of users.

Types of Users.

The most common division is End-Users and DP-professionals. But is it precise enough ?

Not in my opinion; within the group called end-users there is considerable variety. It is obvious that the information needs of a clerical worker differ from those of top-level management. And in between those two extremes there is middle-management with their need for computer power to tailor information on business issues. They also prepare strategic information for top level management.

So we have a more or less passive end-user who uses the computer for operational tasks and the active end-user who is involved in the processing of tactical information and preparation of strategic data.

Top-level management is different because they have two major concerns about data processing: first, the reliability and accuracy of the data they get presented from lower levels in the organisation; second, the cost of data processing.


An important issue for top-level management is to promote and sponsor DP activities, one of these items should be the use of 4GL.

Within the group of DP professionals one can identify a variety of users. Most of them are common functions in your environment so I will describe them only briefly and comment on the relevance to them of 4GL.

Operators will gain in their work, through fewer interruptions due to failure of badly written programs. Programmers will have more job satisfaction because they can emphasize the creative part of their job. They get rid of the necessity of writing repetitive coding. Analysts get rid of extensive documentation work and can concentrate on their mission, in the design stage. They will no longer be frustrated by end-users, complaining that they got something different to what they expected.

The main benefit for DP managers in the use of 4GL, will be their ability to make better estimates on software development cost, as well as reducing the cost of maintenance.

All these advantages I have more or less randomly chosen, without going into much depth.

	HP 3000	ALO4/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Kinds of Tools.

Glancing at 4GL does not exclude the necessity to have a look at available tools. The questions we have to ask ourselves are: can we not fulfil the needs of our (end-)users with tools that are already in operation ? Do we really need something new, and does the implementation improve the quality of our services ?

This brings us back to the question of the improvement of the human interface to (existing) tools. Related to the heading of my paper, we have to concentrate on the tools that already exist in the border zone of the end-user and the DP centre.

We can point at tools for end-users with limited DP experience, like report generators intended for that particular purpose.

More experienced users will wish to extract data from the central processing unit, and process that information on their PC. This requires skills in the use of packages like Lotus, Framework, etc.

Some very progressive users are involved in design and implementation of low-level complexity, where they contribute to screen and report design. This means that more analyst skill must be present. In these cases it is just the magnitude of the project that determines whether this is a DP project, or an end-user-built application. It is where the job of the DP professional starts.

Programmers also can use report writers to solve the needs of end-users. But, for the fulfilment of their job, many other utilities are available e.g. source code generators, editors, re-usable coding, copy-libs, etc.

Tools specifically intended to be used by the analyst are not very common. Most are suited for text processing; only a few of them really support the design task. In Holland we have a tool available, called PRISMA, and it is used within my company. It does not take away the creative part of the task, but it ensures the completeness of the design. An interface to the work of the programmer is not included.

Last, but not least, we should have a look in the office of the DP manager. Is he aided by any tools ? If he has convinced the members of his team to use a structured method for system design, his work will benefit from unambiguous reporting on project development. To schedule projects, perhaps he will use a package like Pertpac, etc.

Drawing a conclusion from the forgoing, we see that available tools are not integrated into the development cycle and that they are also inconsistent to the (end-)user.

The promises of 4GL.

The four major goals covered by 4GL are: faster program development, reduction of the number of errors in programs, reduction of maintenance cost, and portability. In preparation for this paper I read extensively on this subject, and in virtually all cases the same objectives are expressed, or even promised, but aren't they all related to the work of DP professionals? Is there no end-user involved in systems development ? I will investigate his interests first.

Through the eyes of an end-user, most applications have a very long life cycle, 10 to 15 years not being exceptional. This must ring a bell with DP professionals because it outlasts the lifecycle of hardware and operating systems. Most commercial applications reflect the business policies of an organisation. And, because most corporations have invested substantially in securing their marketing niche, it is unlikely that they would diversify or segment so far that completely new applications are necessary.

A second item related to end-users is, that although business policies do not change, applications are very volatile. They are continually enhanced, modified and converted. These revisions are caused by changes in the organisation resulting from growth, corporate revisions, user requirements, technological advances, and changes in government regulations .

Due to the fact that today end-users are more mature (and assertive), and benefit from better education, more change requests can be expected from them. A DP centre is not a business goal in itself, it is a means to an end. The promise to an end-user must be the ease of change.

How many times is an end-user confronted with a statement like; "Your request is feasible and we will implement it, but it will cost you X thousand dollars and it will take N months" ? After several times the user will not come anymore and he will be very disappointed in the effectiveness of his DP centre. I think it's no longer feasible to confront the end-user with rigid systems.

We, as DP professionals, have realized, and the end-users will agree, that existing applications are the backbone of a corporation, and the effort to maintain these applications is enormous.

Another promise to fulfil by 4GL: faster delivery of systems - a great benefit for end-users. Faster implementation gives a corporation advantages in its struggle with the competition, it will be ready for the fray. The problem with the foregoing issue is that it cannot be expressed in bold figures, but the spin-off in quality is there.

Another major benefit of 4GL to an end-user will be that he gets what he saw on the screen. Integration of screen report and design facilities will astonish him.

And what are the promises for the DP professional ?

- * More creative work and less maintenance.
- * Better planning and less stress in project development.
- * Better documentation and less tedious work.
- * Shorter development cycle and more variety in projects.
- * Projects divided into smaller sections and smaller project groups.
- * More days off and less overtime.
- * Phased implementation of projects and end-users who are more satisfied.
- * More end-user involvement and shared responsibilities.
- * Better documentation guidelines and more structured work.

I think all these arguments are true and I will come back to these issues in the next chapter of this paper.

Differences in Application Development.

To describe this clearly I will divide the development process into distinct stages, according to a standard model used throughout my company.

1. On the first stage is the Information Plan which is derived from the Business Plan and Corporate Strategies. The Information Plan gives a description of all the information streams necessary for the achievement of the business goals. Part of the Information Plan is the Computerisation Plan that contains all those information streams suitable for computerisation.

On this stage there is hardly any difference whether a 4GL is in use or not. Tools and aids more geared toward these stages are project control, budgeting and word processing.

2. The second stage is encountered when an information stream is computerised, and a project starts.

We can distinguish here:

2.1. The Feasibility Study researching profitability and cost-effectiveness of a project. In this stage the present situation must be analysed. As a result, requirements and needs are selected for the future system. The next step is to design the improved organisation and to describe the functional specifications of the system. A proposal for hardware and software completes this stage. This is where creative thinking about business issues takes place. The major functions to be computerized are described in a report. Limited information can be stored in a design dictionary.

2.2. The Requirement Definition demands a detailed analysis of the existing organisation and information streams. During this analysis it is possible to identify all data elements, and, functionally group them in collections related to screens and reports e.g. order entry. As a result of this, a Data Model can be built. From this the physical lay-out of the database can be constructed. In this stage the input and output also need to be defined. Selection of hardware and conversion requirements are established.

Differences

4GL	3rd generation and before
* Graphic capabilities	* Hand drawn structured schemas
* Machine generated reports	* Man-dependant reporting
* Involvement of end-users	* Specialist skills needed for writing down the concept
* Tool limits violation of design rules	* Manual consistency checking
* Result aimed at end-users	* Too much DP jargon

3. The technical realization consists of two consecutive steps aimed at the translation of the system into a computer hardware, and software environment.

3.1. Technical Design means transforming the logical design to the technical lay-out at an application, transaction and program level. All necessary online guidance is identified and a detailed description of calculation rules and screening of data elements is produced. For all screens and reports, proposals are produced.

Differences

- | 4GL | 3rd generation and before |
|---|---|
| * Automatic generation of screen and report proposals | * Tedious handwritten documentation |
| * Automatic reporting | * All paperwork |
| * Prototyping with the end-user | |
| * Involvement of end-user in design of help-screens and calculation functions | * End-user confronted with computer-jargon |
| * Ease-of-change in design | * Troublesome changes in design |
| * Ability for more proposals | * One-and-only design |
| * Prevention of errors by emphasizing on design stage | * Errors remain in design and are costly to redress |
| * End-user documentation | * Documentation aimed for technical design |

3.2. Programming and testing. In this phase all technical specifications are converted into machine-readable code. Adoption of a method for program design is essential to prevent spaghetti-coded programs. All specifications must be unambiguous and leave no room for misinterpretation. A major task, also in this stage, is testing. Volumes could be written about how to test thoroughly, few do it well.

Differences

- | 4GL | 3rd generation and before |
|--|---|
| * Only writing processes | * All I/O and processing must be written from scratch |
| * Structure is prescribed by tool | * Structure to be invented by programmer (or tool) |
| * User manuals always reflect current state of development automatically | * End-user documentation must be written after approval of the system by the end-user |
| * Efficient testing | * Tedious testing |

4. Maintenance of software can be split into either minor changes, or the reconstruction of an application. Minor changes must be made directly in the source code, and, of course the documentation updated. Reconstruction of an application will happen when major changes are required, or if the effort to implement minor changes becomes too great. In general this is considered as a complete new project and handled as such.

Differences

- | 4GL | 3rd generation and before |
|---|------------------------------|
| * Changes in the functional description | * Search for where, how, why |
| * Up-to-date documentation | * Weak documentation |
| * Quick (not dirty) | * Time consuming |
| * Instant changes | * (Hidden) Backlog |
| * Early decision on rebuild | * Continuous maintenance |



I have tried to show in brief the differences between 4GL and earlier generations at the various stages of software development. Of course, these arguments are arbitrary, but it gives you an idea of what will change when a 4GL is implemented. One final remark I have to make at this point is with respect to project control and documentation. A good 4GL should be able to help you a lot in this field, because we expect that project size and elapsed time will be reduced. This makes projects more manageable. The other item is documentation; so far all computerisation projects have suffered in this important area. A good 4GL must support the documentation work extensively.

The Changing Role of End-users.

If we look at the history of systems development, we see that end-user involvement in the past was very limited. Steering committees, project groups, etc were staffed by top-level management and DP professionals. For the end-user it was only permissible to have a say in the logical design stage.

Why did this happen ?

In my opinion, it was because in the 1970's and early 80's, you needed extensive know-how about operating system facilities, system software packages and peculiarities of disks and tapes. You had to think mathematically and be able to express solutions in a structured way.

The result; computer freaks were asked to write down end-user wishes, and the end-user was completely surprised by the results.


Nowadays, using a 4GL, we are able to protect the end-user from all the strange habits of computers; the approach becomes more "end-user friendly". At the beginning of this paper I spoke of the "Ivory Tower" and I know it exists, because, until now, system development has been a task for DP professionals. The only problem is that they know everything of computers but rarely have any commercial background.

Let's face the fact, most of the people present here are technicians. To build systems for end-users means an understanding of what drives the business. Should this mean that every end-user must be allowed to build his own system ? And is he able to do so ? Is it of benefit for the corporation he is working for ?

These and a lot of other questions have to be answered. In my opinion it is the responsibility of top level management to support, control, and be involved in DP operations in the widest sense. The success of DP operations starts with the support and involvement of end-users.

The first pay-off of solid management support, is involvement of end-users in systems that will be developed. The details of a system proposal will better represent the business environment.

The second pay-off is that the business will have a fresh picture of itself. There will be less political arguing about how things "ought" to be.

	HP 3000	AL04/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

If top-level management does not agree that it must contribute, I urge that we change its attitude. This can be done by arranging courses or seminars where all computer jargon should be forbidden.

So far I have spoken about top level management, but what about the position of clerical workers and middle-management? What are their concerns, and how is their role changing in the DP environment?

End-users will take a greater part in the system development cycle, due to the availability of end-user-friendly development tools, prototyping techniques and the growing presence of PC's. Having a say in the development cycle also means sharing responsibilities in establishing new and rebuilt applications. This can only be done by expanding knowledge about the development of systems.

So, not only the DP professional should be trained in the usage of a 4GL, but also the end-user must be aware of the possibilities and limitations. This means that the selection of a 4GL is a task not only for the DP department, but end-users should also be involved in the evaluation of potential new tools of strategic importance.

I think end-users are nowadays better educated, and have suffered enough in the past from bad experiences, that they are now in a position where they deserve to have an influence on the decisions that change their role. Together with these end-users, we have to realise ourselves that a 4GL is not a panacea. If we don't play by organisational and procedural rules we will create chaos. If we disobey these rules we will only be busy with creating chaos in a higher gear.

Critical Success Factors.

There is no recipe for success with the implementation of a 4GL; the only contribution I can make, is to present a list of pitfalls, gathered from experiences and literature. First I shall try to describe those factors which are related to end-users, and later I will explore the factors in the DP environment.

- * It is impossible, without sufficient training to introduce a 4GL to end-users. If you expect them to share responsibilities it is obvious that they need skills to bear this. The aim of the training must be to describe functions of an organisation.
- * Avoid a tool where the end-user gets involved in the usage of programming-like statements. All language statements that exceed the level of, for example, PC-DOS commands, must be discarded.
- * Stimulate the usage of the tool and credit the end-user for his productivity increase. A reward is earlier implementation or preferential treatment with maintenance.
- * User-oriented documentation must accompany the product. It must protect the end-user from computer jargon. The documentation must be written in a way that stimulates the end-user to use it. Look for clear, well-explained examples, regular updates, and the promise of news, tricks and tips.



- * Protect the end-user from the fear of blocking or even stopping the computer, destruction of vital data or excessive charges for computer cycles. If the end-users' system crashes, it must not harm the other applications. If things like this happen, end-users will be frightened to use the product.

And what are the success factors for DP professionals ?

- * After education, their skills will be on the 4GL level. This means that they have to emphasize more the functionality of a system, rather than the technical aspects. Probably they need education on that subject as well.
- * More emphasis on new application development instead of boring maintenance. Most DP professionals agree that their laboriously acquired skills deserve better use. It is more challenging to offer alternatives to end-users, than to practice the habits from the past.
- * 4GL's are in use in numerous corporations, and they all benefit from it in their day to day work. This means that you are not so much of a pioneer, with all the attendant risks. It will work in your environment as well if you adopt these tools company-wide.
- * Not only adopt the tool, but incorporate it in your development cycle. Include new procedures, and planning and project control techniques. The creation of a new development "bible" may be necessary.
- * With the proliferation of 4GL we can share responsibilities with end-users. Computer development work becomes more common to your fellow workers in the organisation.

What of the cost-benefits of 4GL with application development ?

Just be pessimistic and assume that it will increase the overall productivity with a factor of 2. Tangible benefits can be ascribed to reduced development time and direct labor savings. This also applies to benefits from earlier implementation of systems and the reduced maintenance cost. Then consider the cost to acquire a 4GL. Based on business "yardsticks", I have no doubt you will decide to buy.

If we are aware of the critical success factors, we also have to be aware of the objections which are commonly made against the employment of 4GL. I will give them briefly and am sure you will find the right answer in your situation:

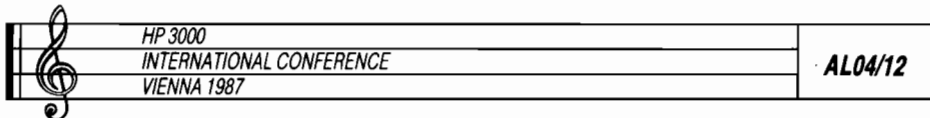
- conservatives don't believe new methods really work
- they are not useful for every application
- DP managers are afraid of erosion of their authority
- application building by end-users creates chaos
- tool is hardware (in)dependent
- it erodes my job
- discounts years of experience.

Experiences with ARTESSA/3000.

I could not have presented this paper if I had only theoretical knowledge. I'm pleased to say, I was able to build experience by using ARTESSA/3000, a 4GL developed by my employer RAET BV in Holland. We have sold ARTESSA/3000 several times and use the tool extensively, for both turnkey and System House projects for users, and for the development of our commercially oriented packages. All conventional COBOL writing is replaced by the complete and extensive use of ARTESSA/3000. Returning to the title of my paper, I shall describe the experiences in the development and maintenance cycle of software and the influence on the communication with end-users.

- * The Feasibility Study is in general not much influenced by the use of ARTESSA/3000. The separate parts of a system are described in a report. A cost/benefit analysis and an estimate for the cost of consecutive steps is added.
- * The Requirements Definition is done using the PRISMA technique. Creation of schemas is an iterative process in close cooperation with the end-user. It clarifies the information streams and makes the division in subsystems easier. The analyst uses ARTESSA/3000 to describe the data elements and the end-user describes the "help" information. The most important screens and reports are proposed by ARTESSA/3000. To clarify the requirements of an end-user, it is frequently useful to jointly develop prototypes in this stage. We have done this in various projects with great success.
- * The Technical Design stage is performed by building-up the Application System consisting of menus and programs. The analyst selects the building-blocks necessary for the construction of the system. The end-user describes help information and algorithms for all appropriate fields. The calculations are organised into a library of reusable computations.
- * Programming is done by coding the calculation rules. This coding is added to the selected building blocks. Then the system is tested by the analyst and programmer. All coding written in the ARTESSA/3000 language is syntactically correct and accepted by the COBOL compiler. Routines written in COBOL or any other HP-supported language are, of course, subject to the scrutiny of the appropriate language compiler.
- * Acceptance testing is performed by the end-user. Errors (which can only occur in processes) are corrected, quickly and easily.
- * Maintenance is a matter of selecting the right building block in the construct and amending it. If a record lay-out is changed, all influenced programs can be determined from the extensive cross-references held in the inventory of ARTESSA/3000. It is our experience that changes of this type are not a burden with systems generated by ARTESSA/3000.

Finally, I want to comment that all stages of development are supported with clear, system-generated, documentation.

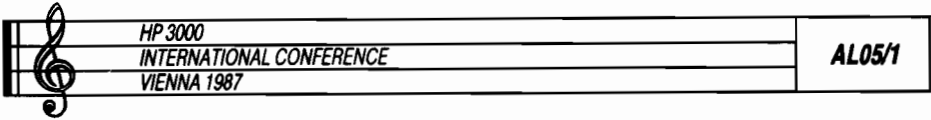


This is not a commercial presentation of ARTESSA/3000, so if you would like to have more detailed information, I invite you to visit booth 319 and I will be pleased to answer all your questions.

Biographical notes:

Hans van der Leeuw is Product Manager ARTESSA/3000, with RAET Software Products in Holland. His experience with computers started in 1960, and since then he has worked in various DP Management positions. He has also worked as a Consultant in multi-national feasibility studies.

Approaches on the content of this paper are welcomed at P.O. Box 4077, 6803 EB Arnhem, Holland.



4GL - The Controversy Rages On

Karen Heater
Infocentre Ltd.
6303 Airport Road
Suite 300
Mississauga, Ontario
L4V 1R8

Introduction

I truly believe that the question which we are asking ourselves regarding 4th Generation languages is quickly changing from "Should I?" to "Which one and how do I make it successful?".

After speaking at a number of Interex Conferences and local area user groups on various topics surrounding 4GLs, the feedback from you, the Data Processing Professional of today and the future, seems to bear out this belief.

The goal of this paper is to assist you in selecting and successfully implementing 4th Generation Development Software. The paper will touch on the strengths of 4GLs, their potential dangers, a few suggestions on how best to approach your product evaluations and lastly some things that must change of you are to be successful.

I hope that at its conclusion you will feel more equipped to make the best decision possible for both you and your organization.

A Rising Need

The emergence of 4th Generation Languages and their associated alternative development methodologies grew out of a clear and definite need arising in the data processing industry.

Let me take a moment to explain -

A recent computer industry forecast predicted that the number of installed computers would increase by a factor of ten in the next ten years.

A Computerworld survey indicated that the number of computerized applications in existing data processing departments is growing at a rate of 45% per year.

In Computerworld it was also estimated that \$10.00 was the average cost of one debugged computer instruction.

Last year, North America produced more computers than programmers and science graduates combined.

And finally, it is also generally accepted that the average backlog of applications in North American DP departments is between three and four years. Visible backlog, that is.


It becomes blatantly apparent that some drastic changes are necessary in our approach to the development of application software if we are to be able to deal successfully with existing and future growth.

The problem we are faced with then is how are we to be successful, not only to survive the next decade, but to conquer it, without the necessity of creating programmers out of every person in our organization.

Enter 4GLs

For a start it seems logical to address the productivity of the DP Professional. Helping the DP Professional to develop systems more quickly would definitely have significant impact on current growth, the backlog and the influx of future needs.

What is needed then are tools to assist the DP Professional in doing his or her job. Tools that would take care of most of the mundane and repetitive aspects of developing production application systems. Tools that "understand" what is involved in the fundamental functional areas of an application; i.e. data storage structures, menu driven end-user interface, online data collection screens, online and batch reporting, transaction processing, application security, etc. and is able to assist the DP Professional in pulling it all together.

	HP 3000	AL05/3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Traditional 3rd Generation Languages are unquestionably logic driven. The constructs of the language are put together to form logic to perform screen handling, reporting, etc. These languages do not easily lend themselves to a transformation whereby they become development assistants. Virtually all of the "smarts" of application development must reside with the individual because the language itself does not truly provide for this.

4th Generation Languages have been loosely defined by various individuals on various occasions as being "products which produce results in one-tenth the time of Cobol". In order to ensure this productivity increase, these languages are designed to be more action than logic driven. In other words, when developing an application, one uses the constructs of the language to state what is needed i.e. screen layouts, report layouts etc., not how logically to perform the task. The language itself handles the fundamentals of interaction with the data storage structures and dictates a certain standard interface for the user.

These languages by nature, assist the DP Professional by providing intelligent defaults throughout the development cycle and imposing certain application design standards. (We will discuss the benefits of these standards a little later.)

What have emerged then are a new breed of development language which when used properly (this cannot be emphasized too much) will significantly affect the individual DP Professional's productivity.


But Wait, There is more

The crux of the problem which appears to be facing us for the decade to come seems to lie in the "development bottleneck", our inability to implement solutions to satisfy all the user needs in a timely manner. The failure to provide these solutions begins the steady build up of that infamous 'backlog'.

Backlog

I'd like to digress for a moment a discuss this 'backlog' that is so often found in discussions of the need to improve productivity.

Backlog results from a development bottleneck. Backlog is catastrophic both from a morale standpoint and its effect on the progress of computerization within an organization.

	HP 3000	AL05/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

We have often heard of backlog described as being comprised of two parts - the visible and the invisible. Visible backlog is readily identifiable, we can describe and count the requests for enhancements or new development that we are just not capable of getting to right now. Visible backlog is difficult enough. We all know that an in-tray that continues to grow despite our dedicated efforts to empty it, is not the most uplifting of experiences. It would seem enough to resign ourselves to a certain quantity of visible of backlog, as we do a certain level of paper in our in-tray. Unfortunately there is still the problem of Invisible backlog.


Invisible backlog is the wealth of applications that users stop asking for. And further down the road, it is the applications that users stop thinking about.

The fact that users stop asking may appear a blessing. We may even attempt to rationalize that if they stop thinking about it, it relieves them of their own frustrations. But users are the people who run the daily operations of our organizations. They build products, sell them, collect payment for them and as a result the business makes profit or some variation on this theme. It is they who can tell us how to help them to perform their function more efficiently so that business can grow. The loss of their input, their ideas, over the long term is devastating.

Without a doubt the 4th Generation Languages that exist today are capable of dramatically improving productivity. You do not need to take my word for this. Above and beyond an intellectual discussion of their merits and potential, we as vendors are always more than happy to provide you with an extensive customer reference list of HP3000 and 4GL users who will discuss their experiences with you.

Certainly, improving each DP Professional's productivity by providing products such as 4GLs is a valid start and possibly even the most far reaching solution. There are two other areas though that should be reviewed in order to completely address the issue of maximizing productivity in an attempt to successfully meet our needs now and in the future; Packaged Software and End-user Computing.

Packaged Software - Ah, the plight of packaged software. At one time Turnkey Systems abounded. A basic package, a little customization, a friendly unassuming minicomputer and voila, a smooth entry into the world of automation. As educational institutions added "Computer Science" to their curriculum, a wealth of programmer/analysts converged on the marketplace and there appeared a growing "computer literacy" across the nation.

	HP 3000	AL05/5
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

What resulted was a feeling that it would be more cost effective to employ one's own programmers to develop software that truly would meet the unique needs of the organization. The movement to bring development and maintenance in-house was on.

The growing popularity of productivity software, namely the 4GL has in the past caused many people to ponder the question, "if inhouse DP Professionals can produce applications at least 10 times faster now then why bother to purchase packaged software at all?".

Well the answer to that is quite simple, but it is still surprising the number of organizations that purchase a 4GL and unnecessarily and most importantly, unwisely, begin to re-invent the wheel over and over again.

More realistically, 4GLs and Packaged Software make a beautiful team. Your packages often become the building blocks of your system while the 4GL allows for the customization, enhancement and integration.

End-user Computing - I can imagine that the same shiver of apprehension that just went up my spine may very well have gone up your spine too.

The enormous growth in popularity and sophistication of the micro-computer workstation has made this an area worth discussion.

Many requests from users that traditionally required one-off customized reports can now be adequately handled by the user, through a combination of electronic spread sheet and graphics software available for the PC. The major area of concern seems to be the accessibility of corporate data to the users of this PC based software. This is a whole area of discussion unto itself and it is more sensible here to direct you to the many good papers available that discuss this topic in-depth. In fact, I have one myself should you be interested.

To recap then, an approach to providing solutions that combines

- Productivity Software (4GLs)
- Packaged Software
- End-user Computing

will put you well on your way into the next decade successfully.

Watching Out

I hope at this point you feel comfortable with the role of the 4GL. It certainly appears that their use will be the most effective solution in our approach to data processing in the next decade.

The 4GL is new and as such there is still much to learn about how best to take advantage of its power. From the experiences of those who have already made the plunge, come the following potential dangers that it would be wise to be aware of so that your implementation can be as smooth and successful as possible.

The real danger of a 4GL is in trying to turn it into your entire solution, i.e. using the 4GL to re-invent the wheel rather than investigating the availability of packaged software.


A 4GL is not necessarily, and most often not an end-user tool. Suddenly providing an end-user with access to your 300,000 customer records so that he or she can produce for themselves that "very simple" report can become a disaster. Do they know what an 'Image Key' is and how one can just provide a value for it rather than serially reading through each customer record looking for the one they want? One request from the backlog may be removed but someone has to handle the potentially serious problem of system performance degradation.

Another danger of the 4GL lies in the DP Professionals approach to its use. A 4GL is dramatically different by nature than a 3GL. The basic approach or methodology is different. 'Translating' an application from a 3GL to a 4GL will not yield the most efficient implementation of the application. It is important to re-think the application, as if you were developing it for the first time, using the 4GL.

4GLs inherently and by design will cause an increase in the speed of development of production systems. Two things to watch out for here are; impact on data and the implementation bottleneck. A little later under 'managing the 4GL' we will discuss how to minimize both of these.

Before we move on to a brighter subject, that of the benefits associated with the use of a 4GL, there is one last area that should be reviewed, that of system performance.

It is not necessary to accept significant performance degradation in return for your productivity gains. There are a number of extremely good 4GLs on the market today and one in particular, that is known for its "Speed" on the system. I'd like to take a moment to discuss a common misconception that seems to be prevalent with respect to performance before we move on.

	HP 3000	AL057
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Performance

I have often heard it asked "If I buy this 4GL will I need to buy more memory and disc in order to use it?". There are really two answers to this question, 'not necessarily' and 'possibly'. Confused? - let me explain.

A good 4GL does not create the need immediately for more power or resources. But, a good 4GL will probably create the need for more resources within 12 months of its purchase.

4th Generation applications can and do perform well in production environments. It is not typically the addition of a 4th Generation production application that chews up system resources, it is the constant use by a staff of DP Professionals over a reasonable period, let's say one year, that does.

When you consider the quantity of maintenance and new development that will have occurred it is obvious that the HP3000 is within one year, handling what would previously have been 3-5 years of work. It is this volume of new systems and improvements to existing ones that has caused the need for more power and resources.


The awareness of the impact of the dramatic increase in the volume of applications on your current system is the most crucial issue here.

The Benefits

A 4GL is a powerful tool that must be managed properly in order that it be an effective one for your organization. When managed properly there are significant benefits to be accrued by the DP Professional, the user and the organization as a whole. It is because of these benefits that each and every HP3000 DP department should be taking a serious look at the acquisition of a 4GL.

* Savings, Savings, Savings. Increasing the productivity of existing DP Professionals could potentially alleviate the need for additional personnel. Applications both backlogged and new requests, can be put into production at least 10 times more quickly giving users access to systems to better assist them in their daily activities. The more productive the users, the more productive the business.

* Controlling Maintenance. The nature of a 4GL being predominantly action rather than logic driven, means that the code itself is much less and more concise. It is easier and less costly to maintain 1000 lines of code than 10000. As applications are put into production faster, there is a better likelihood that they will meet the immediate needs of the user. As a result, less time is needed to perform maintenance tasks that are just bringing a production system up-to-date with the current needs of the user. The cause of these dramatic reductions in maintenance is the use

	HP 3000	AL05/8
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

of the prototyping or protocycling approach to systems development when using a 4GL.

* **Morale.** It is without a doubt most satisfying to work in an environment where you are perceived to be doing your job well. By putting solutions into place quickly for users, the DP Professional feels accomplishment and users are much more confident about the DP Department's ability to respond and satisfy their needs.

* **Standards.** As we discussed earlier, the 4GL assists the DP Professional by imposing a variety of design standards. Two applications developed by different people, with a good 4GL, will look very similar if not almost identical to the user. The system will function in the same manner i.e. menus, screens, prompting, message display. The time needed for a user to become productive on a new system is minimized. The DP Professional can then be spending the time previously spent on training the user, on new development. These standards manifest themselves also in the code itself. Not only do the two applications function the same but the code will be very similar. Readability and standardization of code account for enormous savings in development and maintenance time.

It is obvious that despite the potential disaster areas in using a 4GL there are tremendous benefits in using the right one, properly. I'd like to devote the rest of this paper to the two issues, Choosing the right one and Managing its use effectively.

Choosing the right one

There are really three different levels on which your short list of 4GLs should be evaluated -

1. Functionality / Performance
2. Comfort
3. User acceptance.

Functionality / Performance - It is at this first level that the different 4GLs need to be evaluated to determine if they are capable of being used to put together the type of production applications you need, that will execute with what you consider acceptable performance.

Acquiring and using a demonstration tape and visiting or speaking with reference sites are probably the two most fundamental tasks which should be undertaken at this stage. Both are time consuming but there is really no alternative.

Comfort - Each 4GL has its own personality. When designed the individuals responsible had their own ideas about what the 4GL should do will and how the average DP Professional would use it.

There will probably be one style or personality that you and your personnel will feel the most comfortable with. This is very important because ones comfort and confidence in using the 4GL will determine the degree of productivity improvement that will be attained.

It is through the use of the trial tape that you can assess the effectiveness of the vendor's support services, another very important aspect of the 'comfort' in the use of the 4GL as there is a large learning curve which everyone will have to go through.

User Acceptance - If the 4GL does everything you want it to, the DP Professionals love it but the users can't stand how the applications look, we have not necessarily accomplished anything.

Remembering that often the bulk of defaults or standards that the 4GL imposes will be in the end-user interface area, this could indeed become a problem. During the trial tape period is a good time to put together something which a number of key users can get their hands on and evaluate. A trip to a reference site along with the DP Personnel would probably also be beneficial.

Some Tips

Just a few things to look for that can often be overlooked when evaluating, a trial tape but are quite fundamental to a successful implementation -

How well does the 4GL integrate with your existing environment, i.e. programs, files, security, etc.


How easily does the 4GL allow you to access subroutines in 3GLs. (Don't forget that Fortran pricing routine developed by someone in 1968 that you must use).

What tools / aids does the 4GL provide to assist with implementation. i.e. end-user documentation (Don't forget the development bottleneck will quickly be replaced with the implementation bottleneck)

Effectively Managing the 4GL

Choosing what appears to be the best 4GL for your organization is an important and crucial first step. Now all you have to do is get it implemented and then the new systems that will be developed, implemented.

Having made the purchasing decision I'll assume your selling job to management, users and DP Personnel has been successful. You now have a product and a lot of people have a lot of expectations and probably some unspoken reservations.

	HP 3000	AL05/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Management

Let's start with management. Technological advances that allow machines to perform significantly better are commonplace and easily accepted. Management tends to be far more sceptical about claims of dramatic improvements in people productivity. Their stop watches will be out and running soon after the purchase order leaves the building.

An important aspect of effective management of a 4GL is the setting of expectations. Management should be clear on the fact it will take a number of development cycles before the true productivity gains can be realised. Have them put away the stop watches for at least the first 6 months, giving the DP organization time to settle in.

Be conservative. It is much more fun to exceed expectations and be here later.

The Users

Earlier in the paper 'Prototyping' was mentioned. Briefly, the nature of the 4GL lends itself well to a development methodology where the users and DP Professionals interact on a regular and on-going basis to work an application from an initial prototype to a fully functional production system. This methodology requires the commitment by knowledgeable users to the development process.


Again we are speaking of expectations. Constant dialogue between DP and users was one of the first things to go as a result of the lengthy development cycle typical with 3 GLs. This breakdown in communication is the real culprit behind the devastating problem of invisible backlog.

It will take time and practice to put this dialogue back into place, to re-open the lines of communication. The commitment must be there. If it is not, applications will be developed more quickly, but they will still not meet the real needs of the user.

The DP Department and Professionals

Finally, the DP Department and the DP Professional. Being asked to play with a couple of 4GLs as a short term project is one thing but being given a 4GL and being told to "chuck" Cobol is quite another.

"Keeping an open mind" as an accepted philosophy will be your only chance of survival.

	HP 3000	AL05/11
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

During the evaluation cycle the following areas that pertain to the DP department and/or each DP Professional should be addressed:

1. Data security and control
2. Development Methodology
3. Fear.

Data Security and Control - As the development cycle speeds up and your system becomes populated with new applications and all of their associated files, it is extremely important to have databases and files organized and documented. If they are not, you could find yourself in a reactionary state dealing with one disaster after another. The 4GL will have effectively take control of you.

Development Methodology - 4GLs are not best used with the development methodologies that we have spent years mastering using 3GLs. Frozen specifications, long development cycles, masses of program documentation and maintenance of analysis and design bugs are a thing of the past. Prototyping / Protocycling is the development methodology that is required with a 4GL. I would recommend that you take the time to review all current literature on the subject of prototyping.


The point to be made here is that your current methodologies will not work with your new 4GL. A full understanding of and adherence to a new methodology must occur early on with the 4GL.

Fear - In reality the average DP Professional has spent his or her formal education and years of practical experience refining the skills of application design and development using a 3rd Generation Language. They are competent and confident in their abilities. The implementation of a 4GL means that there is a new learning curve to be addressed and many of the skills, so finely refined are not applicable in this new approach.

There is potentially fear for ones job and alleviating that, fear of success in ones jobs with the new tool.

Effective use of a 4GL does require an emphasis on some skills that were not as important before - namely communication skills, as the bottleneck shifts from development to implementation.

It is important for DP management to take a good look at the people in their department and individually evaluate the level of re-training each person will require and prepare the appropriate education.

	HP 3000	AL05/12
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Starting off with the straight forward applications will go a long way in assisting everyone in managing the new learning curve, building confidence, nailing down development guidelines and bringing users back into the development cycle.


In Summary

4GLs are serious and valuable new tools to assist in managing the requirements of the Data Processing Department in the decade to come.

Much thought and care must be taken in their selection and implementation - to ensure their success with your organization.

Most important is the setting of realistic expectations and the understanding that a 4GL will inherently change the nature of how DP professionals do their job.

If managed effectively the acquisition of a 4GL can mean tremendous savings and a much needed breath of fresh air.

	HP 3000	AL06/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Migrating PowerHouse Applications to New Machine Environments

Paul Elder
Jim Sinclair
COGNOS INCORPORATED
3755 Riverside Drive
Ottawa, Ontario
Canada
K1G 3N3

Abstract

One normally changes computer environments in order to reduce cost, increase performance, increase reliability, increase capacity or some combination of these. The new architecture machines from HP promise to deliver all of these.

Unfortunately, when one changes computer environments the old software does not necessarily work in the new environment. This paper describes how to migrate your PowerHouse applications from the HP3000 to HP's new architecture machines (MPE/XL machines) with as little pain as possible.

Some constructs in PowerHouse relate to specific features of MPE, IMAGE or KSAM on the HP3000. Although similar features may be available on the new machine the means of accessing them may be slightly different. This paper describes what constructs in PowerHouse are non-portable and what to do about them.

Migrating the software to a new environment is only half the question when moving PowerHouse applications. Migrating the data to the new machine also presents a problem. Migrating the data can be as simple as doing a STORE on your HP3000 and a RESTORE on your new machine, or as difficult as taking data apart bit by bit and putting it back together again in a new format. This paper discusses how to recognize when you have a data conversion problem, and how to go about getting the data converted properly.



1. Migrating Your PowerHouse Applications

1.1 What Do I Do First?

When you receive your new MPE/XL machine contact your local Cognos representative. Provide your rep with:

1. The model and serial number of the machine you are upgrading from (if you are upgrading), and
2. The model number and serial number of the new machine you have received.

Your Cognos representative will arrange for you to receive a copy of Compatibility Mode PowerHouse licensed to run on your machine, a copy of Native Mode PowerHouse licensed to run on your machine, and the upgrade documentation set. These copies of PowerHouse can then be installed on your MPE/XL machine. It is not recommended that you attempt to install earlier versions of PowerHouse under MPE/XL, you are not licensed to use them on MPE/XL, and they are not guaranteed to work.

1.2 The Recommended Migration Strategy

When migrating your PowerHouse applications you do not have to do them all at once provided you follow a few simple rules.

1. DO NOT recompile a dictionary with Native Mode QDD until all of the PowerHouse modules (screens, requests, and reports) that use that dictionary have been recompiled with Native Mode PowerHouse. This is because Native Mode PowerHouse products will run off of a 5.01 created QDD dictionary, a Compatibility Mode created QDD dictionary, or a Native Mode created QDD dictionary, but Compatibility Mode PowerHouse products will only run off a 5.01 created QDD dictionary or a Compatibility Mode created QDD dictionary.
2. DO NOT convert your data files until all of the PowerHouse applications that use the data files have been converted to Native Mode.
3. DO NOT try to run Compatibility Mode compiled modules with Native Mode products.
4. DO NOT try to run Native Mode compiled modules with Compatibility Mode products.

Whether you convert all of your applications at once, or one at a time, your conversion must flow through the following five steps:

1. Prepare your application for migration under MPE/V.
2. Convert to running Compatibility Mode PowerHouse. (The brave can skip this step).
3. Convert to running Native Mode PowerHouse.
4. Convert your MPE, KSAM and PowerHouse subfiles to use IEEE floating point formats. (If you don't store floating point data you can skip this step).
5. Convert your TurboIMAGE databases to HPIMAGE databases. (Floating point conversions in databases must be done concurrently with changing databases).

2. Program Conversion

2.1 Preparation on MPE/V

Before bringing your application up on your MPE/XL machine, there are a number of things that must be done first on your MPE/V machine.

1. Upgrade your PowerHouse application to use 5.01 PowerHouse. There is no direct upgrade path to MPE/XL from earlier versions of PowerHouse.
2. If you have not done so already, upgrade your IMAGE databases to TurboIMAGE. There is no support for pre-TurboIMAGE databases under MPE/XL.
3. STORE all of your UDCs necessary for running PowerHouse.
4. STORE all data files, MPE, KSAM, permanent PowerHouse Subfiles and TurboIMAGE databases.
5. STORE all source and object PowerHouse code for your dictionaries, screens, reports, and requests.

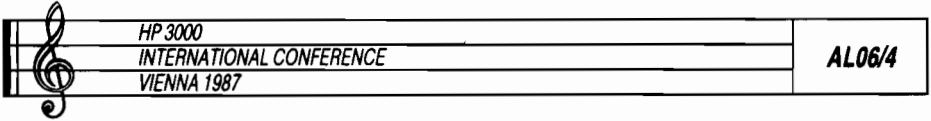
2.2 Conversion to Compatibility Mode PowerHouse

Once your MPE/XL machine is up and running satisfactorily you can migrate your applications to Compatibility Mode by carrying out the following steps:


1. Install Compatibility Mode PowerHouse.
2. RESTORE all the files previously STORED on your MPE/V machine.
3. Modify your UDCs to point to the Compatibility Mode versions of PowerHouse installed in step 1 above.
4. Set all the necessary UDCs, at the system, account, and user level.
5. If you have hard coded the program names of the PowerHouse products (such as QUIZ.DD501E.COGNOS) in any of your application modules, you will have to go through and edit them to point to the new Compatibility Mode products and then recompile. The recommended way to do this is to point to the products in the group CURRENT.COGNOS. This way you will not have to change your source code again when you upgrade to a new version (such as Native Mode PowerHouse) in the future.
6. You are ready to run Compatibility Mode PowerHouse on your new MPE/XL machine. Note that there is no need to recompile your dictionary or any of your screens, reports, or requests, other than the changes you made in step 5 above. Test the new application. If all goes well, continue on to convert to Native Mode PowerHouse. If you discover problems, contact Cognos Technical Support.

2.3 Conversion to Native Mode PowerHouse

1. Install Native Mode PowerHouse.
2. If you skipped conversion to Compatibility Mode PowerHouse, RESTORE all the files previously STORED on your MPE/V machine.



3. Change your UDCs to point to the Native Mode versions of the PowerHouse products.
4. Set all the necessary UDCs, at the system, account, and user level.
5. Recompile your PowerHouse modules using the Native Mode versions of the products. DO NOT recompile your dictionary with Native Mode QDD unless you have converted all applications that use the dictionary.
6. Test the new application. If all goes well, continue on to convert your data. If you discover problems, contact Cognos Technical Support.

	HP 3000	AL06/5
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

3. Data Conversion

3.1 Conversion of MPE Files

All forms of MPE files supported in PowerHouse 5.01 under MPE/V are also supported on MPE/XL. Permanent MPE files can be moved by doing a STORE on your MPE/V system and a RESTORE on your MPE/XL system. The only data types that will need attention are floating point numbers.

3.2 Conversion of KSAM Files

For first release of MPE/XL, KSAM files will be supported on MPE/XL through KSAM INTRINSICS running in Compatibility Mode. This means that all accesses to KSAM files will result in a switch to Compatibility Mode to execute the INTRINSIC. This leaves KSAM applications with less room for performance improvements since the file system (KSAM) is not in Native Mode. KSAM files can be moved by doing a STORE on your MPE/V system and a RESTORE on your MPE/XL system. The only data types that will need attention are the floating point numbers.

3.3 Conversion of Powerhouse Subfiles

All Subfile formats supported by PowerHouse 5.01 under MPE/V are also supported on MPE/XL. Subfiles can be moved by doing a STORE on your MPE/V system and a RESTORE on your MPE/XL system. The only data types that will need attention are floating point numbers.

3.4 Conversion of IMAGE Files

The central issue of almost every migration to the MPE/XL machines will be the choice of the IMAGE database system to use. Native Mode PowerHouse supports all the various forms of IMAGE available. Let's review the key features of each.

3.4.1 Types of IMAGE

The MPE/XL machines support both TurboIMAGE, and HP's new network database HP Image. Accompanying HP Image is TurboWindow, which provides access to HP Image as if it were TurboIMAGE.

TurboIMAGE. TurboIMAGE is a Compatibility Mode product. Everything should be identical to the way it was on MPE/V systems. PowerHouse applications require no changes to work with TurboIMAGE.

TurboWindow. TurboWindow allows access to HP Image databases as if they were TurboImage databases. Although there are a few TurboIMAGE features not supported by TurboWindow, PowerHouse does not use any of these. PowerHouse applications require no modifications to run with TurboWindow.

Note that TurboWindow will not permit utilization of most of the new HP Image features, such as relation sets. The objective of TurboWindow is to provide support for TurboIMAGE while taking advantage of the improved performance of the Native Mode database system.

HP Image. HP Image is a Native Mode network database system. HP Image features which affect PowerHouse are:

- HP Image is an IMAGE look-alike built on a relational core. The nature of a relational core means that:

All HP Image Intrinsic calls (HPI calls), such as HPIGET, HPIPUT, etc., must be within a transaction defined by HPIBEGIN and HPIEND calls. Changes made in a transaction are only committed and made visible to other database users when the transaction is ended.

- Once a transaction is committed (via HPIEND), all context is lost. This includes chain and current record, as well as all locks.
- HP Image provides automatic page level locking. Accessing a page via HPIGET or HPIFIND causes a shared read lock on the page. Changing a page causes an exclusive page lock. Locks are only released when a transaction is ended.
- Changes can be made to pages already having shared read locks, but the transaction changing the page cannot commit until all read locks are released.
- Explicit locks are allowed at the database and dataset levels. HPILOCK is only allowed within a transaction. Locks may only be released when the transaction is ended.
- All HPI calls except HPILOCK wait unconditionally until all required locks are granted. HPILOCK allows both conditional and unconditional locking.
- Transactions may be rolled back with HPIUNDO at any time prior to the HPIEND call. All changes made in the transaction are undone. HPIUNDO ends the transaction and releases all locks.
- If HP Image detects that a deadlock would be created by an HPI call, then the transaction containing that call will be rolled back and ended.
- HP Image supports a new dataset type, the RELATION set. A RELATION set incorporates the features of MANUAL MASTER and DETAIL sets. In addition, generic access is allowed on RELATION set key items.
- Each HP Image database belongs to a Database Environment (DBE). The DBE is the unit of transaction logging and recovery. Transactions may include several databases from the same DBE.
- HP Image has no mechanism for simultaneously committing transactions. Thus, changes to data in several DBEs cannot be guaranteed to jointly commit or jointly fail.
- HP Image allows up to eight sort items on each DETAIL and RELATION set search item. Extended sort items are not supported, and sort items may be positioned anywhere in the dataset.

3.4.2 Supporting HP Image

The preceding list of HP Image differences has meant that PowerHouse must handle HP Image differently from TurboIMAGE. The next sections will detail how HP Image is supported.

Dictionary Support. The FILE statement in QDD has been expanded to support HP Image datasets. In addition, the KEY syntax on the ITEM statement has been expanded to cope with multiple sort items. The FILE syntax for HP Image datasets is:

```
FILE name ORGANIZATION  { AUTOMATIC [MASTER]      }
                        { DETAIL                    }
                        { MASTER                    }
                        { RELATION                   }
```

```
OF databasename IN dbenvironment
[ PASSWORD string ]
[ TYPE HPIMAGE ]
[ OPEN datasetname ]
[ CAPACITY n ]
[ COPYLIB copylibkey ]
[ CREATE/NOCREATE ]
[ DESCRIPTION string [,] string . . . ]
[ read/write lists ]
```

Note that a database environment must be specified in addition to the database name.

The ITEM statement for HP Image search items is:

```
ITEM element  [ storagetype ]
              [ CREATE ]
              [ [UNIQUE/REPEATING] [PRIMARY] KEY ]
              [ LINKS TO file ]
              [ SORTED ON item [ON item] . . . ]
```

Reading and Concurrency. The automatic locking performed by HP Image can lead to severe losses of concurrency in on-line systems if care is not taken to limit transaction size. Of particular concern are what might be called read transactions. That is, a transaction that only involves database reads. QUIZ reports and the QUICK FIND procedure are the primary users of read transactions in PowerHouse. QTP will also use read transactions when there are no outputs to HP Image.

The obvious method of placing the entire read transaction within an HPIBEGIN/HPIEND pair can drastically affect database concurrency. A modest QUIZ report could end up holding read locks on large parts of a database, effectively preventing any updating activity. Even worse, QUICK FIND and SELECT modes could have much the same affect.

To circumvent these problems, PowerHouse implements read transactions as a series of much shorter HP Image transactions. Before moving to a new transaction, the database context must be saved, and then re-established in the new transaction. Figures 1 and 2 show how this is done.

Starting a chained read:

```
HPIBEGIN
HPIFIND

repeat
  HPIGET mode 5
save current record number
HPIEND
```

Continuing a chained read:

```
HPIBEGIN
HPIFIND
HPIGET mode 4 with saved record number

repeat
  HPIGET mode 5
save current record number
HPIEND
```

Figure 1. Multiple transaction chained read in HP Image

Starting a sequential read:

```
HPIBEGIN
repeat
  HPIGET mode 2
save current record number
HPIEND
```

Continuing a sequential read:

```
HPIBEGIN
HPIGET mode 4 with saved record number

repeat
  HPIGET mode 2
save current record number
HPIEND
```

Figure 2. Multiple transaction sequential reads in HP Image

The optimal number of HPIGETs done within a single transaction is a balance between concurrency and optimal access time. One HPIGET per transaction would provide maximum concurrency. On the other hand, there is little value in locking a page only to immediately relock it. Also, the cost of the three or four extra HPI calls could easily become prohibitive. At this point, no good data is available to indicate what the best solution is.

QTP Locking and Transactions. The recommended QTP locking strategies involve opening all files and applying database or dataset locks that are held for the duration of the RUN or REQUEST. The nature of HP Image locks means that the lock duration must also be the transaction duration. Thus, REQUEST level locking means that each request will be contained in a single HPIBEGIN/HPIEND pair. Similarly, RUN level locking implies a single, RUN level transaction.

RUN and REQUEST level transactions have both advantages and disadvantages. On the positive side, they can be used to guarantee that no changes are committed unless the RUN or REQUEST succeeds. On the other hand, an automatic rollback of 500,000 updates could take a long time.

Finally, a word about UPDATE level locking. When using this locking method, QTP does the following:

- During INPUT PHASE, a read transaction similar to QUIZ's is used to improve database concurrency.
- During the OUTPUT PHASE, QTP defines transactions about each individual output action. For example, changing an existing record would involve calls to HPIBEGIN, HPILOCK, HPIGET (reread, and checksum), HPIUPDATE, and then HPIEND. This is the same sequence of calls that would be used with TurboIMAGE.

While UPDATE level locking allows concurrent database access, it does not provide complete consistency, nor can it take full advantage of the HP Image transaction rollback facility. Performance is also adversely affected by UPDATE level locking.

QUICK Transactions. QUICK makes use of both read and update transactions. Read transactions are used in FIND and SELECT modes to retrieve each screen load of data. Read transactions are also used when performing edit lookups. Update transactions are used in the UPDATE procedure.

In FIND or SELECT mode, QUICK begins a new read transaction every time a new screen load of data is requested. The transaction is terminated before any terminal reads are done. QUICK does not maintain locks or transactions across terminal reads! QUICK saves the current context and then restores it again when the next screen load of data is requested. As with other file types, changed records are reread and checksummed as part of the PUT verb processing. Automatic read locks are not used to hold a lock on data being viewed.

UPDATE Procedure. The function of QUICK's UPDATE procedure is to commit all data changes to file. If the update fails, QUICK will rollback any changes made by the failed UPDATE procedure. Thus, the QUICK UPDATE procedure is the natural scope for an update transaction. QUICK begins the UPDATE procedure with an HPIBEGIN call. It is terminated with either an HPIEND or an HPIUNDO call. It is terminated with either an HPIEND or an HPIUNDO call. The following table relates the HP Image calls used to the verbs in the UPDATE procedure.



QUICK Verb	HPI Action
STARTLOG	Marks database for inclusion in update transaction.
LOCK	HPIBEGIN to start transaction if not already started. HPILOCK.
PUT	HPIBEGIN to start transaction if not already started. Re-read with HPIGET and recompute checksum. If checksums are the same HPIPUT, HPIDELETE, or HPIUPDATE
STOPLOG	HPIEND
UNLOCK	ignored
Any error	HPIUNDO

TABLE 1. HP Image calls corresponding to verbs in QUICK UPDATE procedure.

QUICK Rollback. When dealing with HP Image databases within the same Database Environment, QUICK does not maintain rollback information. Instead, QUICK relies upon HPIUNDO to do the rollback. Rollback information continues to be kept for other file organizations. QUICK will only commit an HP Image transaction after all updates involving other file organizations have succeeded.

When QUICK must update files from several Database Environments, rollback information must be kept for all DBE's except one. This is because HP Image cannot simultaneously commit several transactions. QUICK will commit all the DBE transactions for which it has kept rollback information after updates to other file organizations have succeeded, but before the last DBE transaction is committed. In this way, QUICK can attempt manual rollbacks on the committed transactions if the last commit should fail.

Coping with Automatic Rollback. When using HP Image either directly or through TurboWindow, PowerHouse must deal with a transaction being rolled back by HP Image itself:

- If a read transaction is rolled back, PowerHouse will retry from the last successful read. After repeated failure, it will give up.
- If HP Image rolls back a QUICK update transaction, the QUICK UPDATE procedure will fail, and result in a full rollback of any changes. The changes will still be on the screen, so that the user can retry the update.
- If an update transaction fails in QTP, further action is controlled by the error actions defined for the request. These allow either for processing to continue at the next transaction or for run/request termination. Under no circumstances will the transaction be retried.

Effects on QUICK Procedure Code. The FIND procedure has an implied HPIBEGIN and HPIEND pair defining a read transaction enveloping it.

The UPDATE procedure has an implied HPIBEGIN and HPIEND pair defining a read transaction with intent to write enveloping it. STARTLOG verbs encountered within an UPDATE procedure do not start a new transaction unless an explicit STOPLOG verb has been used to close the implied transact. It is VERY STRONGLY recommended that you do not create more than one transaction in your UPDATE PROCEDURE. If you do, roll back recovery may be impossible.

Outside of the FIND and UPDATE procedures GET and PUT verbs will be enveloped with a HPIBEGIN HPIEND pair to form a stand alone transaction, unless explicit STARTLOG and STOPLOG verbs are used to define a transaction.

The net result of these implied HPIBEGIN and HPIEND calls is that most QUICK procedure code should function without change. The one construct that must be reviewed is the use of STARTLOG and STOPLOG verbs because they now define HPI transactions.

3.5 Conversion to IEEE Floating Point Format

If you use floating point numbers you will have to convert to the new IEEE floating point formats.

3.5.1 HP3000/IEEE Floating point types

The table below summarizes the floating point types available on the MPE/XL machines.

Format		Number of Bits		Range		Precision (Digits)
		Exponent	Mantissa	Smallest	Largest	
Float 4	IEEE	8	23	1.4e-45	3.4e38	7.2
	HP3000	9	22	8.6e-78	1.2e77	6.9
Float 8	IEEE	11	52	2.0e-323	7.0e307	15.9
	HP3000	9	54	8.6e-78	1.2e77	16.5

TABLE 2. MPE/XL Floating Point Types

There are restrictions on the usage of the IEEE and HP3000 formats:

- Compatibility Mode programs use HP3000 format floats. There is no supported mechanism for using the IEEE formats.
- Native Mode programs use IEEE format floats. A Native Mode intrinsic is available to convert between the various formats.
- TurboIMAGE only supports HP3000 format floats as item types.
- HP Image only supports IEEE format floats as item types.

The float format used in data files will affect the mix of programs that can access these files, and potentially affect program performance.

3.5.2 PowerHouse Support

Native Mode PowerHouse will support all four floating point formats as item types. Item type syntax has been changed to indicate this. See Figure 1. If IEEE/NONIEEE is not specified, a default format is taken from the dictionary options. A default format may be declared on the OPTIONS statement in the dictionary. If no default format is specified, then IEEE is assumed. Note, however, that when PowerHouse is run with a 5.01 compiled dictionary or a Compatibility Mode compiled dictionary, that the default float format will be NONIEEE.

```

float-item-type is:
  {IEEE   } FLOAT [SIZE {4}]
  {NONIEEE}          {8}

float-format-option is:
  {FLOAT FORMAT {IEEE   }}
  {NONIEEE}
  
```

Figure 3. PowerHouse float-item-type and QDD OPTIONS float-format-option definitions.

3.5.3 Changes In Precision and Range

Converting between floating point types can result in loss of precision or range overflow. There are techniques to detect each of these using PowerHouse.

Loss of Precision. Precision loss can be detected by converting the source type to the target type, and then back to the source type. If the original and final values differ, then precision has been lost. This simple scheme gets slightly complicated because the most precise float format available to Native Mode PowerHouse for internal computations is IEEE float 8. The trick is to use the CHARACTERS function to do the final comparison so that no calculation conversions must be made:

```

;; Precision loss if Original <> Converted

DEFINE Original  CHAR * 8 = CHARACTERS ( original-float )
DEFINE New-type  new-type = original-float
DEFINE Old-type  old-type = New-type
DEFINE Converted CHAR * 8 = CHARACTERS ( Old-type )
  
```

Floating point precisions are shown in Table 1.

Range overflow. Range overflow may be caught by taking advantage of zero being the result of expressions with conversion errors.

```

;; Range overflow if Original <> 0 and Converted =0

DEFINE Original  old-type      = original-float
DEFINE Converted  new-type      = original-float
  
```

When using QTP, remember to include ON CALCULATION ERRORS REPORT on the REQUEST statement.

Approximate floating point ranges are shown in Table 1. Note that if you experience range problems, then you will have difficulty getting QUIZ to report these values, as PowerHouse does not support scientific notation.

3.5.4 Internal Calculation Precision

PowerHouse converts all numeric item values into Float 8 format when performing calculations. Native Mode PowerHouse uses the IEEE Float 8 format. As a result, calculation precision will be slightly less than was provided by HP3000 PowerHouse. This is likely a problem only when calculations require an accuracy close to 16 digits.

3.5.5 Preparing for Conversion

When preparing to convert data files to the new floating point formats several issues must be considered:

Redefinitions. If the floating point item to be converted is part of a redefinition, it is possible that it does not contain valid floating point data.

Figure 2 below shows the record layout for a file that tracks variable names, types and values. Whether FLOAT-VALUE or CHAR-VALUE is used as the value depends on whether VAR-TYPE indicates a numeric or character variable. Clearly FLOAT-VALUE cannot be converted without knowing the value of VAR-TYPE.

Note that files for which several record layouts are defined are another form of item redefinition, and should also be considered when planning conversion.

Conversion Viability. Prior to converting, it is strongly recommended that each file be edited to confirm that existing float values can be stored in the new formats. This is of particular importance when converting from HP3000 FLOAT SIZE 4 to IEEE FLOAT SIZE 4, as the target type has a smaller value range.

RECORD A			
ITEM	VAR-NAME	CHAR	SIZE 20
ITEM	VAR-TYPE	CHAR	SIZE 1
ITEM	VALUE-AREA	CHAR	SIZE 20
REDEFINED BY			
ITEM	FLOAT-VALUE	FLOAT	SIZE 8
END			
REDEFINED BY			
ITEM	CHAR-VALUE	CHAR	SIZE 20
END			

Figure 4. A Record defining a variable name, its type and value.

Sample QTP runs have been included in Appendix A that will edit HP3000 floats and produce a subfile containing all problem records.

3.5.6 Converting

Once convinced of your success, you can start your conversion. Keep the following in mind as you go about your business.

- Only convert one file at a time, if at all possible.
- Remember to edit your data carefully before starting a conversion.
- Do not even think of converting without first making a complete backup!

TurboIMAGE to HP IMAGE. If converting from TurboIMAGE to HP Image, and if all float items are explicitly declared to IMAGE, and if float item values are not context sensitive, then DBMIGRAT may be used. DBMIGRAT is HP's TurboIMAGE to HP IMAGE conversion tool. In the above circumstances, it is likely to be the most effective tool available. Remember that DBMIGRAT converts the whole database in one fell swoop.

QTP and subfiles. This scheme will work for any file. It involves the following steps:

1. Unload the file to be converted into a permanent subfile.
2. Change the dictionary to reflect the new floating point type.
3. Use QUTIL to recreate the file.
4. Reload the file from the saved subfile.

Note that Step 4 will require conditional item final code if float values are context sensitive. Sample QTP runs have been provided in Appendix A to aid in this case. Extreme care must still be taken to guarantee that assignments only occur in the proper context.

One Off programs. A last option is to write a special conversion program in your favourite (next most favourite) programming language. This should not be necessary, but have fun!

3.5.7 Float Conversion Checklist

Here is a list of the things you should do (in the order you should do them) to convert floating point formats.

1. Backup the files you want to convert.
2. Use QTP to run edit checks on the items to be converted. The sample QTP runs provided in Appendix A illustrate each type of edit necessary.
3. If items to be converted are part of redefinitions, analyze these carefully to determine necessary contexts. Do not neglect the other items in the same redefinition!
4. If you are happy with your progress so far, choose the conversion tool:
 - a. Use DBMIGRAT if moving from TurboIMAGE to HP IMAGE, AND if you have no redefinitions, AND if all float items are declared to IMAGE, AND if you want to do all the files in the database at once.
 - b. Use QTP and subfiles to convert one file at a time and to cope with redefinitions. The sample QTP runs should help you with redefinitions.
 - c. Write your own conversion program if you just can't wait to try out a Native Mode compiler.
5. Convert.

4. Additional Conversion Notes

4.1 Process Handling

Process Handling under MPE/XL will be very similar to process handling under MPE/V. Compatibility Mode PowerHouse will be able to call UDCs, other Compatibility Mode programs, and Native Mode programs. Native Mode PowerHouse will be able to call UDCs, MPE/XL script files, Compatibility Mode programs, and other Native Mode programs. All of the differences will be handled within PowerHouse with the COMMAND and PROGRAM statements and the RUN verb. There will be no syntax changes required.

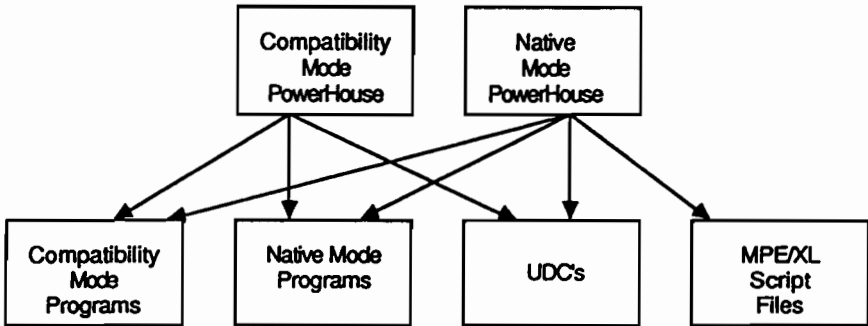


Figure 5. Process Calls Allowed

4.2 Obsolete Datatypes

A number of item data types were declared obsolete in 5.01. Table 3 gives a list of these obsolete datatypes. Compatibility Mode PowerHouse and Native Mode PowerHouse continue to support these obsolete data types, but it is strongly recommended that you move to the new forms.

Item Datatype Equivalents	
Obsolete	New
STRING	CHARACTER
BINARY3	INTEGER SIZE 6
BINARY4	INTEGER SIZE 8
DECIMAL	FREEFORM
DOUBLE	INTEGER SIZE 4
LOGICAL	INTEGER UNSIGNED SIZE 2
LONG	FLOAT SIZE 8
REAL	FLOAT SIZE 4
HPDATE	JDATE
QDATE	PHDATE

Table 3. Obsolete Datatypes

Appendix A - QTP Floating Point Edit and Reload Runs

This first run produces a subfile showing all problem records:

REQUEST Edit-floats ON CALCULATION ERRORS REPORT

ACCESS float-file

:: For each Float 4 conversion add copies of these:

```

::
DEFINE Original4-n    NONIEEE FLOAT SIZE 4 &
                      = float4-n OF float-file

```

```

DEFINE Converted4-n  IEEE    FLOAT SIZE 4 &
                      = float 4-n OF float-file

```

:: For each Float 8 conversion add copies of these:

```

::
DEFINE Original8-n    CHAR * 8 &
                      = CHARACTERS( float8-n OF float-file )

```

```

DEFINE New8-n        IEEE    FLOAT SIZE 8 &
                      = float8-n OF float-file

```

```

DEFINE Old8-n        NONIEEE FLOAT SIZE 8 &
                      = New8-n

```

```

DEFINE Converted8-n  CHAR * 8 &
                      = CHARACTERS( Old8-n )

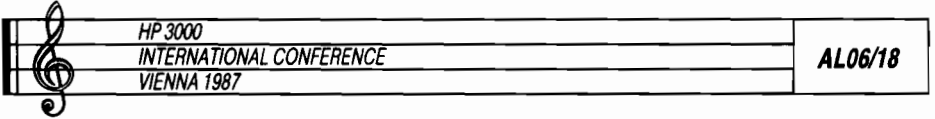
```

:: Create Problems subfile

```

::
SUBFILE <problems> KEEP INCLUDE float-file &
          IF Original4-1 <> 0 and Converted4-1 = 0, &
              or &
              ...
          Original4-n <> 0 and Converted4-n = 0 &
              or &
              ...
          Original8-1 <> Converted8-1 &
              or &
              ...
          Original8-n <> Converted8-n

```



The second run loads and converts a floating point data file from a subfile. Conditional ITEM FINALS are added to handle context sensitive cases:

REQUEST Reload-Floats

ACCESS *reload-subfile

OUTPUT float-file ADD

:: For each context sensitive float item include ITEM FINALS


:: Remember to include ITEM FINALS for other items that

:: overlap with this float, too.

::

ITEM float-n FINAL float-n OF reload-subfile &
IF float-context

ITEM overlap-n FINAL overlap-n OF reload-subfile &
IF NOT float-context

	HP 3000	AL071
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

INFORMATION SYSTEMS PROTOTYPING

Orland Larson
Hewlett-Packard
Cupertino, California

ABSTRACT

One of the most imaginative and successful techniques for clarifying user interfaces and generally improving the productivity and effectiveness of application development is a methodology called INFORMATION SYSTEMS PROTOTYPING.

With waiting time for new applications running into several years and those applications failing to meet the users needs, managers as well as users have been searching for more efficient and effective approaches to systems development.

Prototyping, as an application system design and development methodology, has evolved into a real option for both the MIS professional and the user.

This paper reports on the growing body of knowledge about prototyping. It begins by reviewing the changing role of data processing, the challenges facing the MIS organization, and the traditional approach to application development. It then defines prototyping followed by the step-by-step prototype development process. The advantages and disadvantages, as well as the cost and efficiency of prototyping, will be discussed followed by the essential resources necessary to effectively prototype applications. In conclusion, to illustrate the benefits of prototyping, the speaker will present success stories of systems developed using the prototyping approach.

INTRODUCTION

THE CHANGING ROLE OF DATA PROCESSING

The data processing department has changed dramatically since the 1960s, when application development as well as production jobs were usually run in a batch environment with long turnaround times and out-of-date results.

The 1970s were a period of tremendous improvement for the data processing environment. One of the key developments of that period was the development and use of Data Base Management Systems (DBMS). This provided the basis for on-line, interactive applications. In addition, computers and operating systems provided programmers the capability of developing application programs on-line, while sitting at a terminal and interactively developing, compiling, and testing these applications. The end user was also provided with easy-to-use, on-line inquiry facilities to allow them to access and report on data residing in their data bases. This took some of the load off the programmers and allowed them to concentrate on more complex problems.

During the 1980s, the data base administrator and MIS manager will see increased importance and use of centralized data dictionaries or "centralized repositories of information about the corporate data resources." Simpler and more powerful report writers will be used by the end user and business professional. The programmer will see the trend towards the use of high-level, transaction processing languages, also known as fourth generation languages, to reduce the amount of code required to develop applications. Finally, the tools have been developed to effectively do application prototyping, which will provide benefits to the end user as well as the application programmer and analyst.

Throughout the 70s and 80s, information has become more accurate, reliable, and available, and the end user or business professional is becoming more actively involved in the application development process.

CHALLENGES FACING MIS

One of the MIS manager's major problems is the shortage of EDP specialists. A recent Computerworld article predicted that by 1990 there will be 1/3 of a programmer available for each computer delivered in this country. Software costs are also increasing because people costs are going up and because of the shortage of skilled EDP specialists. The typical MIS manager is experiencing an average of two to five years of application backlog. This doesn't include the "invisible backlog," the needed applications which aren't even requested because of the current known backlog. In addition, another problem facing MIS management is the limited centralized control of information resources.

The programmer/analyst is frustrated by the changeability of users' application requirements (typically, the only thing constant in a user environment is change). A significant amount of programmers' time is spent changing and maintaining users' applications (as much as 60 to 80 percent of their time). Much of the code the programmer generates includes the same type of routines such as error checking, formatting reports, reading files, checking error conditions, data validation, etc. This can become very monotonous or counterproductive for the programmer.

The end user or business professional is frustrated by the limited access to information needed to effectively do his/her day-to-day job. This is especially true for those users who know their company has spent a great deal of money on computer resources and haven't experienced the benefits. The users' business environment is changing dynamically and they feel MIS should keep up with these changes. MIS, on the other hand, is having a difficult time keeping up with these requests for application maintenance because of the backlog of applications and the shortage of EDP specialists. Once the user has "signed off" on an application, he is expected to live with it for a while. He is frustrated when he requests what he thinks is a "simple change" and MIS takes weeks or months to make that change.

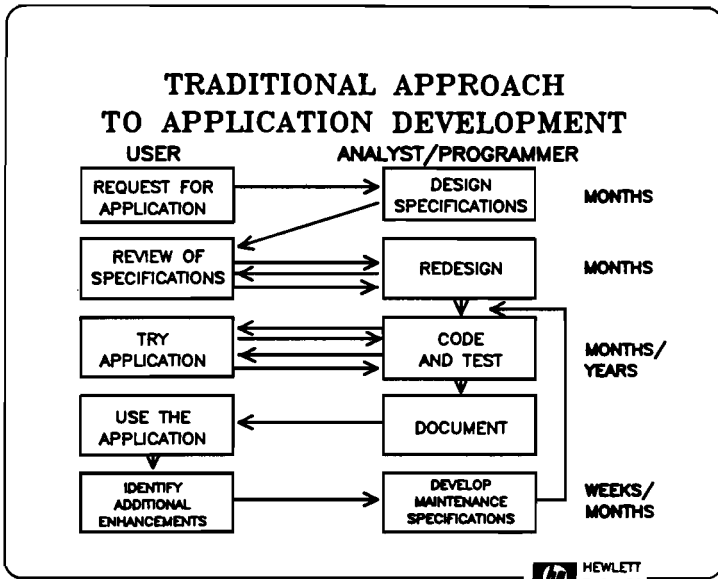
TRADITIONAL APPROACH TO APPLICATION DEVELOPMENT

There are some myths concerning traditional application development:


- Users know exactly what they want
- Users can effectively communicate their needs to MIS
- Users needs never change.

The traditional approach to application development has serious limitations when applied to on-line, interactive information systems that are in a state of constant change and growth. Communications among the user, analyst, programmer, and manager tend to be imprecise, a detailed analysis prolongs the process to the annoyance of the user, and specifications are either ambiguous or too voluminous to read. To compound this problem, the user is often requested to "freeze" his requirements, and subsequent attempts at change are resisted.

Let's review the traditional approach to application development.



- The user first requests an application and then an analyst or programmer is assigned to the application.
- The analyst or programmer takes the oftentimes sketchy user's specifications and designs more complete specifications.
- The user then reviews the analyst's interpretations of his specifications and probably makes additional changes.

	HP 3000	AL07/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

- The analyst redesigns his specifications to adapt to these changes. (By this time, several days, weeks or months have gone by.)
- The user finally approves the specifications, and a team of analysts and programmers are assigned to develop, test and document the application.
- The user finally tries the application. Months or years may have gone by before the user gets his first look at the actual working application.
- The user, of course, will most likely want additional changes or enhancements made to the application. This is called adjusting the application to the "real world".
- Depending on the extent of these changes, additional maintenance specifications may have to be written and these program changes coded, tested and documented.
- The total application development process may take months or years, and the maintenance of these applications may go on forever.


In summary, the traditional approach to application development results in long development times, excessive time spent on maintenance, a multi-year backlog of applications, limited control and access to information, and applications that lack functionality and flexibility and are very difficult to change. The question is: "Can we afford to continue using this approach to application development?"

PROTOTYPE DEFINED

According to Webster's Dictionary, the term prototype has three possible meanings:

- 1) It is an original or model on which something is patterned: an archetype.
- 2) A thing that exhibits the essential features of a later type.
- 3) A standard or typical example.

J. David Naumann and A. Milton Jenkins in a paper on software prototyping (see reference 7) believe that all three descriptions apply to systems development. Systems are developed as patterns or archetypes and are modified or enhanced for later distribution to multiple users. "A thing that exhibits the essential features of a later type" is the most appropriate definition because such prototypes are a first attempt at a design which generally is then extended and enhanced.

	HP 3000	AL07/5
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

ROLES IN THE PROTOTYPING PROCESS

There are two roles to be filled in prototyping -- the user/designer and the systems/builder. These roles are very different from the traditional user and analyst/programmer roles under the traditional approach. The terms "user/designer" and "systems/builder" emphasize these differences and denote the functions of each participant under the prototyping methodology. Remember it is the user who is the designer of the application system and the systems professional who is the builder.

The user/designer initiates the process when he/she conceives of a problem or opportunity that may be solved or exploited by the use of an information system. The user/designer typically must be competent in his/her functional area (many times he/she is a manager) and usually has an overall perspective of the problem and can choose among alternative solutions. However, he/she requires assistance from the MIS organization.

The systems/builder is assigned by the MIS organization to work with the user/designer and is competent in the use of the available prototyping tools and knowledgeable about the organizations data resources.

PROTOTYPING PROCESS

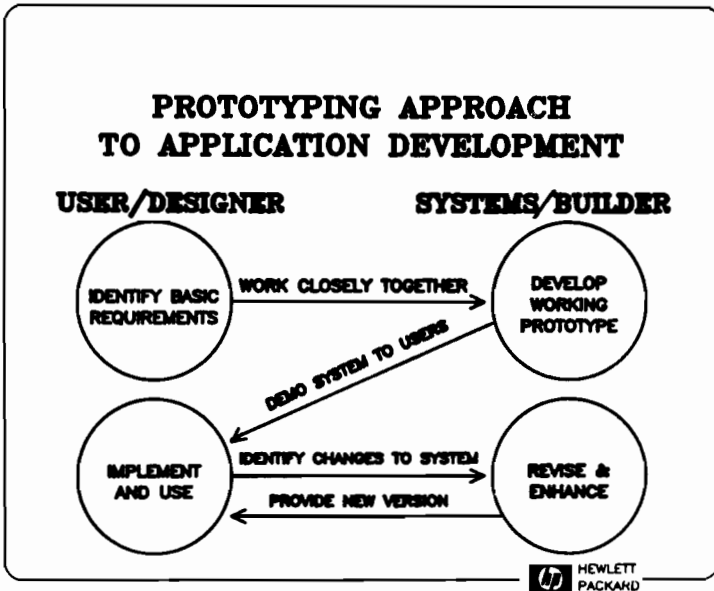
The process of application prototyping is a quick and relatively inexpensive process of developing and testing an application system. It involves the user/designer and the systems/builder working closely to develop the application. It is a live, working system; it is not just an idea on paper. It performs actual work; it does not just simulate that work. It can be used to test assumptions about users' requirements, system design, or perhaps even the logic of a program.

Prototyping is an iterative process. It begins with a simple prototype that performs only a few of the basic functions of a system. It is a trial and error process - build a version of the prototype, use it, evaluate it, then revise it or start over on a new version, and so on. Each version performs more of the desired functions and in an increasingly efficient manner. It may, in fact, become the actual production system. It is a technique that minimizes the dangers of a long formal analysis and increases the likelihood of a successful implementation.

PROTOTYPING METHODOLOGY/MODEL

The prototyping methodology in general, is based on the following proposition: "People can tell you what they don't like about an existing application easier than they can tell you what they think they would like in a future application."

Prototyping an information system can be viewed as a four -step procedure.



Step 1. User/designer identifies the basic information requirements:

- Write a brief, skeleton-like statement that captures the essential features of the information requirements.
- User/designer and systems/builder work closely together.
- Concentrate on users' most basic and essential requirements.
- Define data requirements, report formats, screens, and menus.
- Need not involve lengthy written specifications.
- For larger systems, a design team may need to spend a few weeks preparing a first-effort requirements document.

Step 2. Systems/builder develops the initial prototype:

- Systems/builder takes the notes developed in the user discussions and quickly builds the menus and dialogs.
- A data dictionary would be useful at this time.
- Design and/or define data base and load subset of data.
- Make use of defaults and standard report formats.
- Write required application modules using a 4GL.
- Prototype performs only the most important, identified functions.

Step 3. Users implement and use the prototype to refine requirements:

- Systems/builder demonstrates prototype to small group of users.
- Users gain hands-on experience with application.
- Users are encouraged to make notes of changes they would like made
- Users discuss and prioritize desired changes.

Step 4. Systems/builder revises and enhances the prototype:

- Systems/builder modifies the prototype to correct undesirable or missing features.
- May require modification or redesign of data base, changes to existing programs and/or additional program modules.
- Deliver back to users quickly.

NOTE: Steps 3 and 4 are repeated until the system achieves the requirements of this small group of users. Then either introduce it to a larger group of users for additional requirements or if enough users are satisfied, demo it to management to gain approval for the production system.

WHEN TO USE PROTOTYPING

Richard Canning, the author of the EDP Analyzer (see reference 2), suggests when prototyping should be used.

1. To clarify user requirements:

- Written specs are often incomplete, confusing, and take a static view of requirements.
- It is difficult for an end user to visualize the eventual system, or to describe his/her current requirements.
- It is easier to evaluate a prototype than written specifications.
- Prototyping allows, even encourages, users to change their minds.
- It shortens the development cycle and eliminates most design errors
- It results in less enhancement maintenance and can be used to test the effects of future changes and enhancements.

2. To verify the feasibility of design:

- The performance of the application can be determined more easily.
- The prototype can be used to verify results of a production system.
- The prototype can be created on a minicomputer and then that software prototype may become the specifications for that application which may be developed on a larger mainframe computer.

3. To create a final system:

- Part (or all) of the final version of the prototype may become the production version.
- It is easier to make enhancements, and some parts may be recoded in another language to improve efficiency or functionality.

WHEN NOT TO USE PROTOTYPING

Canning also suggests when prototyping should not be used.

1. When an application requires a standard solution that is already available at a reasonable cost from a software supplier.
2. When you don't have a good understanding of the tools available to prototype.

3. When the organization's data and software resources are not well organized and managed.
4. When MIS management is unwilling to develop a staff of systems builders.
5. When the user/designer is unwilling to invest his/her time in the development of the application system.

POTENTIAL PROBLEMS

One of the initial problems typically encountered is the acceptance of the prototyping methodology by the systems people. This is due to the fact that people naturally tend to resist change. It may also encourage the glossing over of the systems analysis portion of a project. It is not always clear how a large complex system can be divided and then integrated. Initially, it could be difficult to plan the resources required to prototype (people, hardware and software). It may be difficult to keep the systems staff and users abreast of each version of the system. Programmers may tend to become bored after the nth iteration of the prototype. Testing may not be as thorough as desired. It might be difficult to keep documentation on the application up to date because it is so easy to change.

Even with these concerns, prototyping provides a very productive working relationship for the users and the builders. So it behooves all data processing management to learn to use this powerful tool creatively and to manage it effectively.

THE ADVANTAGES OF PROTOTYPING GREATLY OUTWEIGH THE PROBLEMS!

ADVANTAGES OF PROTOTYPING

One of the main advantages of application prototyping is that this methodology provides a capability to quickly respond to a wide variety of user requests. It provides a live, functioning system for user experimentation and accommodates changes in a dynamic user environment. One interesting aspect of this approach is that users are allowed and even encouraged to change their minds about an application's interfaces and reports, which is a very rare occurrence during the traditional approach. Maintenance is viewed right from the beginning as a continuation of the design process. Finally, prototyping provides an effective use of scarce systems/builders. One or a limited number of systems/builders will be required for each prototyping project; and while users are testing one prototype, the systems/builder can be working on another.

COST AND EFFICIENCY

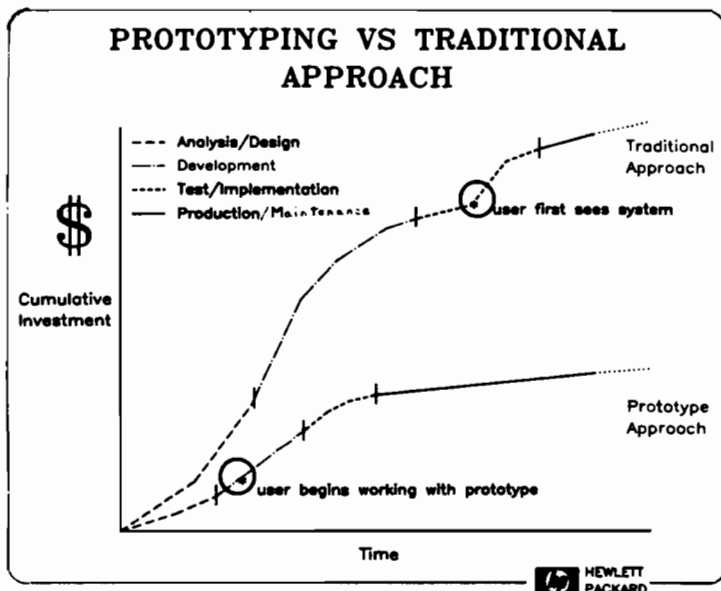
It has been found that there is an order of magnitude decrease in both development cost and time with the prototyping methodology.

It is often difficult to estimate the cost of prototyping an application system because the total costs of development, including maintenance, are usually lumped together. The cost of implementing the initial system is much lower than the traditional approach (typically less than 25%).

However, software prototyping could be expensive in the following ways:

- It requires the use of advanced hardware and software.
- It requires the time of high-level users and experienced systems staff.
- It requires training of the systems staff in the use of prototyping and the associated tools.
- Application run-time efficiency may be compromised.

The main thing to remember is that the main focus of prototyping is not so much efficiency but effectiveness.




ESSENTIAL RESOURCES

The following are the essential resources to effectively do application prototyping:

1. Interactive Systems

- **Hardware and Operating System** - When doing application prototyping, both the builder and the system must respond rapidly to the user's needs. Batch systems do not permit interaction and revision at a human pace. Hardware and associated operating systems tailored to on-line interactive development are ideal for software prototyping.

	HP 3000	AL07/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

2. Data Management Systems

- A Data Base Management System provides the tools for defining, creating, retrieving, manipulating, and controlling the information resources. Prototyping without a DBMS is inconceivable!

3. Data Dictionary

- A Data Dictionary provides standardization of data and file locations and definitions, a cross reference of application programs, and a built-in documentation capability. These are essential to managing the corporate resources and extremely useful when prototyping.

4. Generalized Input and Output Software


- Easy to use data entry, data editing, and screen formatting software are extremely helpful in the application prototyping process to allow the programmer to sit down at a terminal with a user and interactively create the user's screens or menus.
- Powerful, easy-to-use report writer and query languages provide a quick and effective way of retrieving and reporting on data in the system. A report writer that uses default formats from very brief specifications is most useful in the initial prototype.
- A powerful graphics capability can be extremely useful for the display of data in a more meaningful graphical format.

5. Very High Level (Fourth Generation) Languages

- Traditional application development languages such as COBOL may not be well suited for software prototyping because of the amount of code that has to be written before the user sees any results.
- Very powerful fourth generation languages that interface directly to a data dictionary for their data definitions are ideal. One statement in this high level language could realistically replace 20-50 COBOL statements. This reduces the amount of code a programmer has to write and maintain and speeds up the development process.

6. Documentation Aids

- Tools to aid in the maintenance of programs written in a 4GL.
- Tools to aid in maintaining user documentation on-line.

	HP 3000	AL07/11
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

7. Libraries of Reuseable Code

- A library of reusable code to reduce the amount of redundant code a programmer has to write is an important prototyping resource.
- This code could represent commonly used routines made available to programmers.

HEWLETT-PACKARD'S TOOLS FOR PROTOTYPING

Hewlett-Packard is one of the few vendors that supplies the majority of the tools needed to effectively do software prototyping.

* Interactive Systems

- HP 3000 Family of Computers
- MPE Operating System

* Data Management Systems and Associated Utilities

- TurboIMAGE/V
- TurboIMAGE Profiler/V
- TurboIMAGE DBchange/V
- HPSQL/V
- KSAM/V
- MPE files
- HP Silhouette/V
- HP Access Central, HP Access

* Data Dictionary

- Dictionary/V
- HP System Dictionary/V

* Generalized Input/Output Software


- VPLUS/V
- Business Report Writer/V (BRW/V)
- QUERY/V
- REPORT/V
- INFORM/V
- HPEASYCHART
- HP DSG/V

* Very High Level Languages

- TRANSACT/3000

* Documentation Aids

- EDITOR/V
- HPSLATE
- HPWORD
- TDP/V


	HP 3000	AL07/12
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

ADDITIONAL PROTOTYPING TOOLS AVAILABLE FROM HP THIRD PARTY VENDORS

- * **Data Management Systems and Associated Utilities**
 - ADAGER Adager
 - CARESS, INTACT, SILHOUETTE/3000 Carolian Systems International
 - DBACE Snodgrass Consulting
 - DBAUDIT, SUPRTOOL Robelle Consulting Ltd.
 - DB GENERAL, DB-KEY-CHANGE Bradmark Computer Systems
 - DEMGR, IMSAM, OMNIDEX, CAPCHG Dynamic Information Systems, Corp.
 - DBTUNE (Europe Only) HI-COMP
 - FLEXIBASE, BACKCHAT Proactive Systems Ltd.
 - HSC-COPYDB Hawaiian Software Company
 - IMAGINE Technalysis Corporation
 - MINISIS Systemhouse Ltd.
 - MIRAGE (HP 150) Datasoft International
 - PC/IMAGE (HP 150) Advanced Data Services
 - RELATE/3000 CRI, INC.
 - SPEEDEX, SPEEDBASE (HP 150) Infocentre

- * **Generalized Input/Output Software**
 - DATADEX/3000 Dynamic Information Systems, Corp
 - EASYREPORTER Infocentre
 - ENVY, HELPER System Works, Inc.
 - INDEX PLUS Spectrum Solutions
 - MONITOR, MISTRAL (HP 150) Datasoft International
 - PAL DATA REPORTER Gentry
 - PRESENTATION GRAPHICS ARENS
 - PRW/3000 Infotek Systems
 - QUIZ, THE EXPERT, GRAPHICS COGNOS
 - RELATIONAL QUERY/3000 Upland Software
 - SCREEN/3000 RMS Business Systems
 - WHAT-IF CIBAR, Inc.
 - THE WRITE STUFF PROTOS Software Company

- * **Fourth Generation Languages and Utilities**
 - ARTESSA/3000 (Europe only) RAET Software Products
 - CBAS/3000 Comprehensive Systems, Inc.
 - :DBEXPRESS, :DBTRANS SystemsExpress
 - FASTRAN (TRANSACTION Compiler) Performance Software Group
 - FLEXIBLE Sages American Group
 - INSIGHT II Computing Capabilities Corp.
 - LL'SPIRIT Singapore Computer Systems
 - PAL FAMILY GENTRY
 - POWERHOUSE (QUICK) COGNOS
 - PROGSPEC/3000 (COBOL Gen.) Productive Systems
 - PROTOS (COBOL Generator) PROTOS Software Company
 - Q-PLUS Los Altos Software
 - RELATE/3000 APPLICATION BUILDER CRI
 - SPEEDWARE, MICROSPEEDWARE Infocentre
 - SYDAID (Europe only) SYDES
 - THE SYNERGIST Gateway Systems Corp.
 - TODAY BBJ Computers International, Inc.

	HP 3000	AL07/13
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

* Documentation Aids


- | | |
|----------------------------------|-----------------------------------|
| - DOCUMENTOR (Part of SPEEDWARE) | Infocentre |
| - LARC | LARC Computing |
| - QEDIT | Robelle Consulting Ltd. |
| - ROBOT/3000 | Productive Software Systems, Inc. |
| - S/COMPARE | Aldon Computer Group |
| - SPEEDDOC, SPEEDEDIT | Bradford Business Systems, Inc. |
| - TESS/AIDE | Computer Consultants and Serv. Ce |

The preceding lists of HP third-party software are not 100% complete. The majority of the listed software was derived from ads placed in SuperGroup Association Magazine, Interact Magazine and The Chronicle. Please consult the Hewlett-Packard Business Systems Software Solutions catalog (Part # 30000-90251) for additional information.

SUMMARY

Prototyping is truly a "state-of-the-art" way of developing applications.

- Software prototyping promotes an interactive dialogue between the users and the programmer, which results in a system being developed more quickly, and results in an interactive development approach which is friendlier for the end user.
- The prototype provides a live working system for the users to experiment with instead of looking at lengthy specifications.
- The users are provided with an early visualization of the system which allows them to immediately use it.
- The users are allowed and even encouraged to change their minds about user interfaces and reports.
- Maintenance is viewed right from the beginning as a continuous process and because the prototype is usually written in a very high-level language, changes are faster to locate and easier to make.
- Software prototyping results in:
 - * Users who are much more satisfied and involved in the development process.
 - * Systems that meet the user's requirements and are much more effective and useful.
 - * Improved productivity for all those involved in software prototyping: the user/designers and the systems/builders.

	HP 3000	AL07/14
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

REFERENCES

- Boar, Bernard H., Application Prototyping: A Requirements Definition For The 80's, John Wiley & Sons, New York, New York, 1984.
- Canning, Richard G., "Developing Systems By Prototyping," EDP Analyzer (19:9) Canning Publications, Inc., September 1981.
- Jenkins, A. Milton, "Prototyping: A Methodology For The Design and Development of Application Systems," Division of Research, School of Business, Indiana University Discussion Paper #227, April 1983, (41 pages).
- Jenkins, A. Milton and Lauer, W. Thomas, "An Annotated Bibliography on Prototyping," Division of Research, School of Business, Indiana University Discussion Paper #228, April 1983, (25 pages).
- Larson, Orland J., "Software Prototyping - Today's Approach to Application Systems Design and Development," Proceedings 1984 International Meeting HP 3000 IUG, Anaheim, California, February 26 - March 2.
- Martin, James, Application Development Without Programmers, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.
- Naumann, Justus D. and Jenkins, A. Milton, "Prototyping: The New Paradigm for Systems Development," MIS Quarterly, Vol. 6, No. 3, September 1982.
- Naumann, Justus D., and Galletta, Dennis F., "Annotated Bibliography of Prototyping for Information Systems Development," Management Information Systems Research Center Working Paper (MISRC-WP-82-12), September 1982.
- Podolsky, Joseph L., "Horace Builds a Cycle," Datamation, November 1977, pp.162-186.
- Wetherbe, James C., "Systems Development: Heuristic or Prototyping," Computerworld, Vol. 16, No. 7, April 26, 1982.

How to write Structured TRANSACT/3000 Programs

I. Summary

TRANSACT/3000 is a procedural fourth Generation Programming Language! Because it is a procedural programming language, all the rules of structured programming can and should be applied when writing TRANSACT/3000 programs. That paper shows how easy it is to write structured software in TRANSACT/3000 and the resulting benefits.

II. Program Structure

Modern programming languages like PASCAL or MODULA allow to make a difference between local- and global variables (in TRANSACT/3000: items). That feature supports the construction of structured programs, because not only code is part of a procedure, but also data-space and its values. That is in opposite to RPG and COBOL, where you have only one domain for global variables, called WORKING STORAGE. With other words, every procedure can have its own WORKING-STORAGE. The concept of local- and global variables is ideal to write easy understandable and maintainable software, without any side-effects of global items. Before we discuss in details how to construct structured TRANSACT/3000 software, let us define what is a procedure within TRANSACT/3000: A procedure in TRANSACT/3000 is a piece of code, which is executed by a PERFORM statement and has one or more RETURN statements. (According to that definition, a TRANSACT/3000 CALL statement and PROC statement don't call procedures)

II.1 Local- and Global Variables in TRANSACT/3000.

Remark: For the rest of the paper I use the words variable and item as synonyms. TRANSACT/3000 has a LIST-Register, which is basically a stack, where the LIST statement represents the PUSH-statement and the SET(STACK) LIST(*) statements represents the basic version of the POP-statement. In addition it is possible to put the same item several times onto the stack. That remarkable feature is the basic for the concept of local- and global items. In our (structured) software every procedure has following structure:

```

(1)  PROCEDUREXYZ:    <<Procedure Name>>
      LIST BEGIN;

      <<Local Items>>
      LIST L1:
        L2:
        L3;
      .....
      <<Code of the procedure PROCEDUREXYZ>>
      .....
      .....
      .....
      .....

```



- ```
(2) IF (X) < (Y) THEN <<Statement>> ELSE
 PERFORM END;
 <<.....return statement within the procedure>>

(3) PERFORM PROCEDURELMN;
 <<Call of the procedure PROCEDURELMN within the
 procedure PROCEDURE XYZ>>

 <<End of code of the procedure PROCEDUREXYZ>>
(4) PERFORM END; <<End of Procedure>>
```

The statement LIST BEGIN at (1) puts an item (stackmarker) onto the stack when the procedure is activated. The item name BEGIN is a "Reserved" item word and should not be used any more. If you don't like the word BEGIN, take another one. Within PASCAL and other modern programming languages however the word BEGIN is used to indicate the beginning of the local part of a procedure.

During the execution of a TRANSACT/3000 program a lot of BEGIN items will be on the dynamic stack. Even the local items like L1, L2 and L3 can be several times onto the LIST register. The last item pushed onto the LIST register is the item used. The statement at (2) represents an early exit out of the procedure. It will be explained together with the statement at (4). The PERFORM statement at (3) calls a the procedure PROCEDURELMN. That procedure may have its own local variables. All the items of the procedure PROCEDUREXYZ are global items for it. The procedure PROCEDUREXYZ can even call itself with its own local variables. (Direct and indirect recursion is no problem). When the execution of the procedure PROCEDURELMN is finished, the local items of the procedure PROCEDUREXYZ are at the top of the LIST-Register. The statement PERFORM END (4) is used to indicate the end of the procedure. It has two functions: first to delete (pop) all local items from the stack, which have been pushed onto the stack during the execution of the procedure PROCEDUREXYZ; second to go back to the next statement after the PERFORM PROCEDUREXYZ; The return-procedure END looks like this:

- ```
END:
(1) DEFINE(ITEM) BEGIN X(2),OPT:
      DUMMY X(2),OPT;
(2) LIST DUMMY;
(3) SET(STACK) LIST(BEGIN);
(4) SET(STACK) LIST(*);
(5) RETURN(1);
```

The statement at (5) executes a double return to go to the next statement after the (not shown) PERFORM PROCEDUREXYZ; The statement at (3) deletes all local items from the stack. At (4) the local item BEGIN itself is

deleted from the LIST-Register. The LIST DUMMY statement is needed because the SET(STACK) statement works in a very complex way. (For more details see TRANSACT/3000 reference manual)

II.2 Rules for Structured Programming in TRANSACT/3000

1. Think and work with the concept of global- and local items. Global items are items, which are used together with their values in nearly every procedure. In an online oriented software global items can be variables like USERNAME, TERMINALNUMBER, PASSWORD, TIME etc.
2. Keep the number of global items short. Put the items for a data-set only onto the the stack for the procedure or procedure tree where these items are really used. Don't think in COBOL'S WORKING-STORAGE-SECTION!
3. Even if a procedure has no local item, use the LIST BEGIN -- PERFORM END construction. Software is a matter of change and it may have local items after a the next maintenance.
4. Never use local-items within a loop construct like REPEAT ..UNTIL or WHILE...DO. These items are global for that loop.
5. Procedures with parameters within the language definition of TRANSACT/3000 are not (yet) possible. Therefore parameters can only be logical parameters and have (normally) to be global items.
6. If you use the GO TO statement, use it only within the local procedure. Everthing else is dangerous in a lot of respects.

II.3 Advantages and Benefits of Global-and Local Items.

1. You construct easy to understandable software, friendly for maintenancance, software with less or even no side-effects, an important criteria for structured programmms.
2. Procedures have there own variables, a concept of modern programming languages.
3. The stack register will be much more smaller. The problem of "Stack Size" will nearly disapper! (My problem on big TRANSACT/3000 programs is normally not the stacksize, but an overflow of the TRANSACT/3000 compiler tables!)
4. Because the stack tends to be very short, the performance of TRANSACT/3000 programs are similar to COBOL programs.
5. The concept of local- and global items allows easy TRANSACT/3000 (re)segmentation.

III. VPLUS/3000, TRANSACT/3000 and Structured Programming

VPLUS/3000 is very well integrated into TRANSACT/3000. However the TRANSACT/3000 reference manual and HP customer training do



not explain how to use VPLS/3000 in structured environment. The following text will explain how to do it easy and simple.

III.1 Menu -and Presentation structure

Assumptions:

- I. Good online software is menu-driven, not command-driven.
- II. Good online software has a hierarchical menu structure.
- III. Good online software has a standard softkey-layout and standard softkey functions at every screen.

Funktions of softkeys used for the following examples:

F1 = Back to Main-Menu. The Main-Menu is the highest menu in the hierarchical menu structure. If the software shows the hierarchical menu, pressing that softkey has no effect .

F2 = HELP. Whenever that softkey is pressed, a screen depended help information will be displayed. What and how will be discussed later in that section.

F3 = PRINT. Pressing that softkey will send a hardcopy of the screen to a printer.

F4 = REFRESH

F5 = Free (Perhaps always used for confirmation of deletions or next-page funtion)

F6 = Free (Perhaps used for previous page function)

F7 = Free (Perhaps used to start QUERY/3000 for program-testing or "jump" to an application monitor or window)

F8 = Previous Menu or Done. It basically means you are finished with your transaction and you want to go back to the previous menu.

III.2 Structured VPLUS/3000 Procedures

In that chapter we show and explain the procedures to handle VPLUS/3000. The programmer has only to know and use One! procedure. His structured VPLS/3000 procedures will be easy and powerfull to use according to a fourth Generation Programming Language. The demonstrations and explanations following will be in a top-down methode.

Programmers Routine:

SHOWXYZSCREEN:

- (1) LIST BEGIN;
- (2) LIST SOFTKEY:
SCREENNAME;

- (3) LIST <<ITEMS OF SCREEN IF NOT ALREADY ON LIST REGISTER>>
- (4) MOVE (SCREENNAME) = "XYZSCREEN";
 <<NAME OF SCREEN IN ELEMENT>>
- (5) SET(FORM) (SCREENNAME),INIT,CLEAR,LIST=();
- (6) REPEAT
- (7) DO
- (7) PERFORM GETFORM;
- (8) IF (SOFTKEY) = 0 THEN PERFORM XYZACTION;
- (8) DOEND
- (9) UNTIL (SOFTKEY) = 8;<<also 0 if screen is not repeating>>
- (10) PERFORM END; <<EXIT; if Main-Menu>>

If the presentation of the screen is a own procedure, you start with a procedure name and a LIST BEGIN (1). Every screen has always its own local item SOFTKEY (2) to store the value (0..8) of the softkey pressed and the local item SCREENNAME to store the name of the screen (3). One of the biggest advantages of TRANSACT/3000 is that the VPLS/3000 screenname can be a variable in every VPLS/3000 statement like GET(FORM), PUT(FORM), SET(FORM) and UPDATE(FORM). We will see the benefits of that feature later in that section. (5) shows the standard initialisation of a VPLS/3000 screen. From (6) up to (9) we have a REPEAT-LOOP, because the SOFTKEY function F2, F3, F4 can only be handled by a REPEAT-LOOP construction. If the user presses F8 (RETURN or DONE), the loop is stopped at (9) and with the PRFORM END statement at (10) all the local items like SOFTKEY and SCREENNAME will disapper. If that screen is the Main-Menu, you would write an EXIT statement instead of a PERFORM END statement! If the user has pressed the Enter Key (F0), the software would execute the procedure ACTIONXYZ; That label name and the screen name in the move statement at (4) and sometimes the SET(FORM) statement at (5) represents the parameter part of that standard. Every other line of code is the same for every screen. Coming back from the procedure ACTIONXYZ, the programmer can control just by setting or not setting of "0" at (9), if a screen is a repeating screen!

Now the discussion of the procedure GETFORM:

- ```

GETFORM:
LIST BEGIN;

(1) PERFORM SETTIME-DATE;

(2) IF (ERROR-FIELD) <> " "
 THEN

(3) GET(FORM) (SCREENNAME),FKEY=SOFTKEY,
 WINDOW=((ERROR-FIELD),(MESSAGE))
 ELSE
 IF (MESSAGE) <> " "
 THEN

```



- ```
(4)          GET(FORM) (SCREENNAME),FKEY=SOFTKEY,
              WINDOW=(MESSAGE))
              ELSE
(5)          GET(FORM) (SCREENNAME,FKEY=SOFTKEY;
              IF (SOFTKEY) <> 0 THEN PERFORM HANDLESOFTKEYS;
(6)          IF (SOFTKEY) = 1,8,0 <<NO REPEATING SOFTKEYS>>
              THEN
              DO
              MOVE (ERROR-FIELD) = " ";
              MOVE (MESSAGE) = " ";
              PERFORM END;
              DOEND;
(7)          PERFORM END;
```

The procedure SETTIME-DATE (1) is optional. It handles all items which are common in every screen e.g. TIME, DATE, USERNAME etc. Before we go into further details, lets have a look at the WINDOW-option of the VPLS/3000 statements of TRANSACT/3000. Whenever you want to use that option, you have to reset the window afterwards. Otherwise TRANSACT/3000 represents you always the same message in the window. The procedure GETFROM has changed that rule. It only shows ONCE! a message in the window and optional enhance a field in error. The programmer don't have to reset the window. If the programmer wants to show a message in the window, he has to put the error-text into the global item MESSAGE, and if he also wants to enhance a field, he has to put the field-name in error into the global item ERROR-FIELD. At (3), (4), and (5) the procedure GETFORM executes the according GET(FORM) statement. AT (6) the procedure GETFORM resets the window and error-field. However, the resetting is only done, if the softkey pressed was not a repeating one (F2, F3, F4, F7). That "reversed" standard makes the live of a programmer much easier. The functions of the softkeys other than 0 are handled within the procedure HANDLESOFTKEYS:

HANDLESOFTKEYS:
LIST BEGIN;

- ```
(1) IF (SOFTKEY) = 1 THEN <<GO TO MAIN-MENU>>
 DO
 SET(STACK) LIST(ENDOFGLOBAL);
 RETURN(@);
 DOEND;
(2) IF (SOFTKEY) = 2 THEN PERFORM HELP;
(3) IF (SOFTKEY) = 3 THEN PERFORM PRINT;
(4) IF (SOFTKEY) = 4 THEN DISPLAY "REFRESHING...";
(5) IF (SOFTKEY) = 7 THEN PERFORM QUERY;
 PERFORM END;
```

At (1) we see how simple it is to construct a "Go Back to Main-Menu". For implementation the programmer has to follow two rules: 1. After all global items have been put onto the stack, he has to indicate that by loading the item ENDOFGLOBAL onto the stack. When softkey F1 is pressed by the user, the procedure HANDLESOFTKEY deletes all local items from the stack up to the item ENDOFGLOBAL.

2. The "PERFORM ACTION" statement within the REPEAT ..UNTIL Loop of the Main-Menu must be the deepest (not necessarily the first) PERFORM statement. The RETURN(@) statement always continues the execution of the program after the next statement of the deepest perform statement.

A HELP facility (2) can be implemented in a lot of ways. Important is that in the last (youngest) item SCREENNAME the program can find what is the name of the current screen. You can use that item as a parameter for your HELP facility. One idea is to use that value and look in the DESCRIPTION part of that screen in DICTIONARY/3000 and display that text as help-information. Another idea is to create with FORMSPEC.PUB.SYS one or more HELP screens with the name <<SCREENNAME>>H1 up to <<SCREENNAME>>H9.

The PRINT function (3) can be implemented in a lot of ways. One idea is to use the VPLS/3000 VPRINTF intrinsic or, if a integrated terminal printer is available, just execute a DISPLAY "ESC 0" statement or activate the program PSCREEN.PUB.SYS.

The DISPLAY "REFRESHING..." statement at (4) is not really needed. However the DISPLAY statement closes and afterwards reopens the terminal in VPLS/3000 mode and resets all terminal values correctly.

Softkey function F7 can be used very flexible. During the development of an application, that function can activate QUERY via the create- and activate intrinsics to allow fast checks and changes in a data-base. In production the software may call HPDESK or own written application monitor, which allows to "jump" within the applications.

### III. Standards

Every online application has to offer transactions like CREATE, CHANGE, DELETE, FIND and DISPLAY of master-files. One big advantage of a procedural language is that the user can decide how the layout and the logic has to look like in opposite to nonprocedural software like POWERHOUSE, SPEEDWARE, which force you to use a predefined logic. On the other side, there is no need to always reinvent the wheel, that is to write line per line every transaction for every master-file. Instead you should construct once for one master-file Your Layout and Your Logic. For all the other master-files, just replace the screennames and date-set names with the EDITOR, add the special consistency-checks for the according master-file and your software is finished.



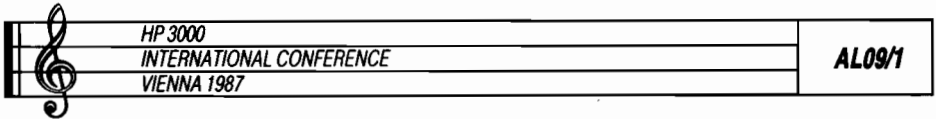
#### IV. Improvements of the Structure for TRANSACT/3000

1. Make TRANSACT/3000 more procedural in the sense of PASCAL or MODULA. Following features should be implemented:
  - a. Real procedures with parameters, BEGIN and END blocks.
  - b. Itemnames, labels and procedures should also be used as parameters beside all other items.
  - c. Definition of TYPES similar to PASCAL should be allowed within TRANSACT/3000 and the DICTIONARY.
  
2. Better Integration of TRANSACT/3000 and DICTIONARY.
  - a. Why is TURBOIMAGE, KSAM and MPE-Files not in the same way and not with the same comfort integrated as VPLS/3000?
  - b. Why can data-set-names and files-names not used in variables similar to VPLS/3000?
  - c. Why does the programmer have to consider how he has to access a file (SERIAL, CHAIN) and what is the key. All the necessary information are already stored in the DICTIONARY! HOW to access should be the job of TRANSACT/3000 with the help of DICTIONARY, only WHAT is the job of the programmer.

About the author: Ewald Maria Mund is an independant consultant. His main subjects are the construction of informationsystems with the help of fourth generation tools. He also works in the field of datacommunications with a strong emphasis of softwareintegration.

Address: Ewald Maria Mund  
Buchholzstrasse 13  
CH-3066 STETTLEN/BERN  
Switzerland

Tel: 031/519839



**The 4th generation environment**

**Rudi Huysmans - Wim Bockstaele  
Sydes NV  
Mechelen - Belgium**

**The 4th Generation Environment has been characterized by the development of powerful generators, query languages and end-user tools.**


**This lecture presents another aspect of the 4th generation world. It will focus on some high level tools for programmers and software analysts and not for end users.**

**An old proverb tells us :**

**"A good workman is known by his tools".**

**In this lecture we will pay more attention to the following tools for programmers:**

- 1. The high level language**
- 2. The application manager**
- 3. The preprocessor and source generator.**

|                                                                                  |                          |        |
|----------------------------------------------------------------------------------|--------------------------|--------|
|  | HP 3000                  | AL09/2 |
|                                                                                  | INTERNATIONAL CONFERENCE |        |
|                                                                                  | VIENNA 1987              |        |

## 1. The high level language

The chief reasons for using a high level language are productivity and debugging speed.

In his book "The mythical man-month", Frederik Brooks has written a collection of thought provoking essays on the management of computer programming projects. These essays drawn from his own experience as a project manager for the IBM System /360 and for OS/360, its operating system. In chapter 8 he has made a comparison between programs written in low level languages and programs writing in high level languages.

Here are the conclusions :

- ° Productivity seems constant in terms of elementary statements, a conclusion that is reasonable in terms of the thought a statement requires and the errors it may include.
- ° Programming productivity may be increased as much as five times when a suitable high level language is used.

I do not have the intention to summarize all the possible features of a high level language. Let us focus on some very useful ones.

### 1.1.1. Definition : scrolling window


A scrolling window (in short : scroll) is a data-window through which we can see or fill in a piece of a table at a time.  
 The data on a scroll-line can be of different types, but the lines itself are of the same kind. The user can run over the entire table (forward or backward), add lines, change and delete.  
 If the table is larger than 80 positions, the user must be able to run through the scroll horizontally.

### 1.1.2. Scroll - statements

A lot of standard statements can be offered to the programmer using scroll :

Here are some examples:

**INSERT** : inserts an element in the scrolling window, above the current line  
**DELETE** : deletes the current scroll line from the scrolling window  
**NEXT** : advances the cursor to the next line  
**PREV** : sets the cursor back to the previous line  
**NEXTP** : advances the cursor to the next page  
**PREVP** : moves the cursor to the next page  
**BEGIN** : puts the cursor on the first field of the first line  
**END** : moves the cursor to the first field of the last line  
**SHOW** : defines the order in which the fields of the scrolling area will be arranged.  
**SORT** : sorts the scroll area in ascending sequence  
**COMPUTE** : performs the specified compute statement on all lines of the scroll area  
**SEARCH** : searches a field value in the given scrolling field

|                                                                                  |                          |        |
|----------------------------------------------------------------------------------|--------------------------|--------|
|  | HP 3000                  | AL09/4 |
|                                                                                  | INTERNATIONAL CONFERENCE |        |
|                                                                                  | VIENNA 1987              |        |

## 1.2. The use of function keys

We define 2 different sorts of function keys : physical function keys and logical ones.

### A. Physical function keys (FKEYS)

The physical function keys are the eight special, physically existing keys (F1, F2, ..) at the head of the keyboard.

To each key a label on the screen is connected.

On most HP screens these function keys are programmable. It is possible to attach a certain function to these function keys.

### B. Pseudo keys

As the number of available function keys is always limited to eight, we can define the so called pseudo keys (PKEYS).

A pseudo key is activated through a 1 or 2 character defined by the programmer and acts like a function key.

### 1.2.2. Action on function keys

#### Enabling physical function keys

```

* FKEY keynumber,routine,string,label *

```

sets up the programmable function key with number "keynumber". The key is loaded with "string" and "label" is displayed in the key label.

'routine' indicates the routine to be executed whenever the function key is hit.

E.g. FKEY 1,deleterout,"%","DELETE"

#### Enabling pseudo keys


```

* PKEY keynumber,routine,string *

```

#### Disabling function keys

```
CLEAR FKEY n
CLEAR PKEY n
CLEAR FKEYS
CLEAR PKEYS
```

|                                                                                  |                          |        |
|----------------------------------------------------------------------------------|--------------------------|--------|
|  | HP 3000                  | AL09/6 |
|                                                                                  | INTERNATIONAL CONFERENCE |        |
|                                                                                  | VIENNA 1987              |        |

### 1.3. The invoke mechanism

An other beautiful feature is usually refered to as: Invoke.

Suppose you are making an invoice on your system. Half way through your screen you fill in the article number 707 and the system answers "This article number doesn't exist".

What to do now.

It's now that the invoke function can be very useful.

How does it work:

You freeze your current screen with its contents. You jump to the article-manipulation-screen. You add the missing article-number with all its data and then you come back, there where you left the invoice screen. The original screen contents are still there and now your article number will be accepted.



## 2. Application supervisory system

### 2.1 The application supervisory system

#### 2.1.1. Definition

The application supervisory system guides the user through his applications without requiring the user to have any knowledge of MPE. It is rather like a coat-stand where the programmer puts his application on.

#### 2.1.2. Concepts

From the point of view of the user, the application supervisory system offers a hierarchical path through the user's application using a set of menus and submenus. An experienced user can enter the taskname and avoid the need to use the tree of options. An inexperienced user, on the other hand, can walk through the menu tree and use a help facility to guide his choice.

The user can run programs, stream jobs, and execute MPE commands without directly using the MPE command Interpreter. The system allows the management of the menus without the need to recompile the application programs.

#### 2.1.3. Standard ways to select a screen.

There are different ways to choose a screen from an application.

- You can walk through a set of menus to find the desired application.
- You can give the final option without having to go down the tree of options. Each form (screen) has a name and the form name can be used to give up a final selection.
- You can invoke a form: the named form is called and displayed on the screen.  
After the form is finished, the current form is re-displayed and continued at the statement following the INVOKE.



#### 2.1.4. Features of a good and ergonomic application supervisory system.

##### The application supervisory system:

- is a utility to handle processes. If you want to add, delete or modify a selection for a given user, you do not have to recompile programs.
- secures user accesses. It identifies the user with his password and the user himself is responsible for handling his own password.
- guides the user's choice, prints or reprints a menu, immediately accepts the user's final options and has a help facility.
- allows parameters such as LIB, INFO, ENTRYPOINT, PARM, \$STDIN, \$STDLIST to be given at run time.
- tests if a selection may be used.
- increases performance because it can verify if a process is still alive and if it is, re-activate it.  
The advantage is that the process is still allocated, and the files are still open, ...  
This is the best way to get a good response time.
- uses a logging system to ensure that a complete history is maintained of all special events.
- gives the opportunity to have all messages in the language of the user.

### 3. An intelligent preprocessor and generator

#### 3.1. Problem description

##### 3.1.1 A waste of time in administration.

User manuals, functional and technical drafts, contracts, reports ... all these documents often contain an important number of standard (or almost standard) paragraphs.

Manuals and functional drafts might have a section with definitions of elementary concepts such as terminal, screen, function keys, menu, ... Contracts and offers often contain warranty formulations, and a description of the sold products.

Documentation should have standard title and subtitle layouts (depending on the printer they will be printed on) ...

A lot of time is wasted looking up the needed paragraphs in earlier documents and copying them into the new one.

Besides, certain data can change in time. (Prices for instance often do not tend to be constant over a long period. From the date of change on, it is necessary to avoid copying the old paragraphs.)

##### 3.1.2. Why programmers need to work so late.


Imagine a devoted programmer who has to implement, say ten, more or less analogous interactive programs. The technical draft requires a persistent standardisation in the use of function keys, screen layouts and the handling of database inconsistencies. These requirements demand a great deal of attention.

Our programmer might develop the first program and then start copying and adapting this source, which will yield him all the others.

This of course seems a reasonable and efficient way of working. Though when testing the seventh program, our programmer finds a bug in one of the copied parts. Is this a newly introduced bug, due to some error in copying ? Or does this bug also exist in some or all of the previously written programs ? In the worst case, all programs have to be retested.

Some time after delivery, our programmer gets a list of requests from the end-users representative, with questions as : "Please replace the function key label 'PRINT' by 'PRINT SCREEN'." "Could you extend the error routine with console logging ?" Ten sources that have to be adapted and retested.

Moreover, can you ever be sure that our devoted programmer has never forgotten to test the Image condition code, after performing a call to the database ?

|                                                                                  |                          |         |
|----------------------------------------------------------------------------------|--------------------------|---------|
|  | HP 3000                  | AL09/10 |
|                                                                                  | INTERNATIONAL CONFERENCE |         |
|                                                                                  | VIENNA 1987              |         |

### 3.2. An outline of the solution.

For the reasons explained above, it might be useful to compile a library made up from standard paragraphs and/or standard coding that can be referenced. This library can be used for drawing up manuals, contracts, reports and programs. Any standard paragraph, (we call it a macro), can be included in the text, simply by referring to its name.

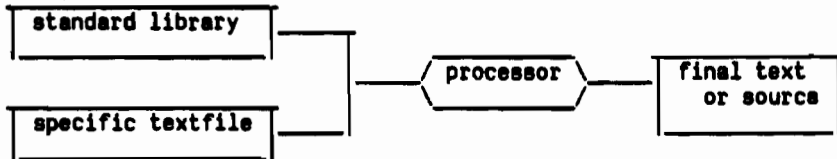
The final text is obtained through expansion of all referenced macronames by means of a 'substitution machine', an intelligent processor with features such as conditional substitution, the use of parameters and nesting of macros. Its userfriendliness can even be improved by an automatic 'forward reference'. This implies that any macro call can precede its definition.

The supplementary processing step may not significantly slow down development time. So a high performance of the substitution machine also is an important requirement.

### 3.3. A functional description of the required functions.

The processor matches a library of standard macros (standard library) and a library of application-related macros (specific textfile) to generate a final text or source. Both libraries are ordinary textfiles and can be composed by the user with any editor, according to some very simple grammatical rules for distinguishing macro definitions, macro calls, parameter settings and conditional substitutions.

Schema :



#### 3.3.1. Basic notions.

We have to agree on some special symbols indicating the beginning and the end of a macroname :

```

starting symbol : <
ending symbol : >
macronames : any characterstring in between a
 starting symbol and an ending symbol
 e.g. <string>
 <unlockdatabase>
 <checkaccountnr>

```

### 3.3.2. Definition of macros.

```

<macro> ::=
|//|
|//|
|//////// macro body //////////|
|//|
|//|
<nextmacro> ::=
|//|
|//////// nextmacro body //////////|
|//|
|//|

```

The end of a macro is given by the beginning of the next one or by the end of the file.

An example :

```

<STROPHE> ::=
Old Mac Donald had a farm, Ia Ia OOh
And on his farm he had some cows
With a mooh, mooh here
and a mooh, mooh there
here a mooh, there a mooh
everywhere a mooh, mooh
Old Mac Donald had a farm.

```

### 3.3.3. Macro calls.

To call a macro, one simply inserts the macroname in the text.

An example :

Here is one strophe of a popular song :

```
<STROPHE>
```

This song has been written during ...

After expansion the macro call <STROPHE> has been replaced by the text from the definition of strophe in paragraph 3.2.


Here is one strophe of a popular song :

```

Old Mac Donald had a farm, Ia Ia OOh
And on his farm he had some cows
With a mooh, mooh here
and a mooh, mooh there
here a mooh, there a mooh
everywhere a mooh, mooh
Old Mac Donald had a farm.

```

This song has been written during ...

	HP 3000	AL09/13
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

### 3.3.4. The use of parameters.

The definition of a macro can contain a number of parameters. A parameter consists of a starting symbol, followed by a number of alphabetical characters.

In this text the exclamation point (!) is considered as the starting symbol.

```
Examples : !PARM
 !DATASET
 !NAME
 !DAY!MONTH!YEAR
```

When calling a macro, values for all parameters can be supplied. Parameter values stand in between special characters like " " or ' '.

We can now rewrite our example, using parameters:

```
<STROPHE>::=
Old Mac !NAME had a farm, Ia Ia OOh
And on his farm he had some !ANIMALS
With a !NOISE, !NOISE here
and a !NOISE, !NOISE there
here a !NOISE, there a !NOISE
everywhere a !NOISE, !NOISE
Old Mac !NAME had a farm.
```

We can define a new macro <SONG>.

```

<SONG> ::=

* VARIATIONS ON OLD MAC DONALD'S THEME *

<STROPHE,NAME='Donald',ANIMALS='cows',NOISE='mooh'>

<STROPHE,NAME='Peter',ANIMALS='pigs',NOISE='oink'>

<STROPHE,NAME='Johnny',ANIMALS='ducks',NOISE='quaok'>

```

After expansion of <SONG>, we get a song with three strophes in which every parameter has been replaced by the given value.

```

* VARIATIONS ON OLD MAC DONALD'S THEME *


Old Mac Donald had a farm, Ia Ia OOh
And on his farm he had some cows
With a mooh, mooh here
and a mooh, mooh there
here a mooh, there a mooh
everywhere a mooh, mooh
Old Mac Donald had a farm.

Old Mac Peter had a farm, Ia Ia OOh
And on his farm he had some pigs
With a oink, oink here
and a oink, oink there
here a oink, there a oink
everywhere a oink, oink
Old Mac Peter had a farm.

Old Mac Johnny had a farm, Ia Ia OOh
And on his farm he had some ducks
With a quaok, quaok here
and a quaok, quaok there
here a quaok, there a quaok
everywhere a quaok, quaok
Old Mac Johnny had a farm.

```

Notice that in the example above, two macros have been nested.

	HP 3000	AL09/15
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

### 3.4. Experiences and results.

A processor as described in the previous section and extended with some other features has been used internally since april 1983, mainly for the development of application programs in Sydaid (a fourth generation language) and for composing functional and technical drafts and user manuals.

#### 3.4.1. General remarks.

As far as programming is concerned, it proved its usefulness on several different domains, such as :

- the standardisation of screen layouts
- the standardisation of function key handling
- the uniformisation of database calls and KSAMfile calls with condition code checks
- special routines such as checks on bankaccount numbers or VAT numbers or other project related control routines.

Although the starting-up period of a project is longer, (all standards have to be implemented first), the use of the processor leads to an important increase in the productivity during the other phases.

For elementary programs, such as a maintenance program on one dataset with read, write, update and delete capacities, the amount of coding can be reduced to less than one third of its original length.

There is not only a gain of time, due to the reduction of the total number of lines to be typed. But, as macros are in fact higher-level commands, also the programmer's way of thinking is simplified and this again implies a higher productivity.

Debugging time decreases significantly, as enormous blocks of coding need to be tested only once, or have already been tested during previous projects.

Maintenance also takes less time, as a lot of modifications can be made on the standard library instead of on each individual program.



### 3.4.2. A case-study.

For a project consisting of 44 interactive screens and 5 batch programs, a Sydaid library with 2824 lines of standard coding has been set up, covering the domains mentioned above.

The length of specific coding totals up to 31082 lines. The total length of all final sources after processing came to 81053, which is a factor 2.6 greater.

In Table 1, some results for different kinds of programs and groups of programs are given.

Table 1.

Programs or groups of programs	number of lines		expansion factor A/B
	before (B)	after (A)	
The whole project . . . . .	31 082	81 053	2.6
An interactive maintenance program on one dataset . . . . .	513	1 813	3.5
A small batch program without database calls . . . . .	142	220	1.5
A simple interactive program without database calls . . . . .	387	1 155	3.0
All interactive programs with database calls . . . . .	22 575	63 471	2.8
All batch programs with database calls . . . . .	5 849	14 054	2.4


As mentioned before, performance of the processor was one of the most important requirements.

Table 2 contains some performance results for various types of programs. The standard library contained 175 macro definitions, one third of it with one or more nested macro calls.

All tests have been performed on a HP3000 series 42, with 2 Mb central memory and a 400 Mb disk with disk caching.

Table 2.

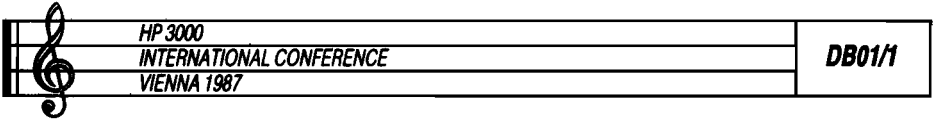
programme	length before	macro calls	length after	CPU time(sec)	elapsed time (sec)
interactive progr with DB calls . .	7493	778	21149	85	118
interactive maintenance progr on one dataset . .	513	74	1813	13	17
batch program without DB calls .	666	9	1017	9	12

	HP 3000	AL09/18
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

### Biography

Rudi Huymans was born in 1953 in Mortsel near Antwerp. After High School he studied civil engineer in the computer science at the Catholic University of Leuven. He joined SYDES in 1982 as a commercial project leader. He is Vice Chairman of the Belgian National Users Group. Off business hours he works as teacher in the computer science.

Wim Bockstaele was born in 1947 in Brussels. Studied graduate in computer science at the NARAFI in Brussels. Joined Sydes in 1984 as system engineer. Specialised in HP systems and specially HP3000. Project leader for SYDAID team in Sydes. Sydaid is a 4th Generation Language developed by Sydes.



## **Data integrity and recovery**

**Robert Bray**

**Carolian Systems International Inc.  
3397 American Drive, Suite 5  
Mississauga, Ontario, Canada  
(416) 673-0400**

## DATA INTEGRITY AND RECOVERY

Database failures and resultant recovery efforts cost HP3000 users thousands of dollars every day in lost processing time and inconvenience. While this paper provides a discussion of IMAGE failure types and various methods of recovery, it also intends to educate the reader as to how some basic database design and implementation procedures can act as preverbial "ounces of prevention" - protecting you and your company from having to needlessly exist and suffer with logically and physically broken databases.

## IMAGE FAILURE MODES

A quick review of IMAGE failure modes reveals that such common occurrences as system failures or hangs, disc media failures, datacomm line failures or application failures can all bring database processing to a screeching halt. Failures can also result in physical or logical damage to the database, with physical failures resulting from having bad data on disc (fileys), broken chain pointers or inconsistent root files. Logical failures can result because of missed updates, puts or deletes or missing delete flags. Whatever the cause, there are several standard and some new ways to repair the damage.

## CLASSIC IMAGE RECOVERY

A standard method of IMAGE recovery is to restore your most recent copy of your database and forward recover using DBRECOV. However, there are some major deficiencies with this method of recovery. First and foremost, the process is extremely time consuming as it keeps users away from productive processing.

Also, DBRECOV uses a technique of recovery known as staging, whereby the restored DB is updated from the log file via staging files. The problem with staging is that large numbers of transactions can be ignored if an "end" is not found, resulting in these transactions not being applied to the database. The result can be a great deal of time and effort spent on forward recovery, with no guarantee that the recovery will be complete.


#### **INTRINSIC LEVEL RECOVERY - ILR**

If a failure occurs during an actual DB intrinsic such as updating, ILR can ensure physical integrity of your database by undoing the intrinsic. The problem with "undoing" is that with IMAGE databases, in some instances the log file and the database may not agree! Improvements made to TurboIMAGE have alleviated this problem.

#### **TURBOIMAGE RECOVERY**

With Turbo, ILR will complete the intrinsic call so that the logfile and the database agree, as opposed to just "undoing" it. Turbo allows you to forward recover with DBRECOV as does IMAGE, but it also allows you to initiate a rollback recovery.

Rollback recovery is a more timely method of recovery as it eliminates the need for a DBrestore and to reapply logged transactions to the database. Rollback recovery allows users to bring their current database up, and back out the last incomplete transactions, while complete transactions are left in place.

	HP 3000	DB01/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The use of ILR and Rollback recovery will generally ensure that more data is recovered than is possible with roll forward techniques. This is due to the fact that ILR with rollback recovery requires physical logging.

Physical logging ensures that the changes to the database are recorded and written to the log file as they occur. This prevents the log record from remaining in memory where they can be lost in the event of a failure.


Despite the time savings that can be realized with Turbo's newer recovery features, neither these or IMAGE recovery procedures are of assistance with another common occurrence that results in logically broken databases - program aborts.

#### **RECOVERING FROM PROGRAM ABORTS**

Programming bugs, user errors and datacomm line failures are just a few of the occurrences that can result in a database becoming logically corrupt. To date, HP3000 sites have had to live with the fact that once their databases have become logically corrupt, that they have to endure this inconvenience until a full recovery procedure can be initiated.

#### **PROBLEMS ASSOCIATED WITH ABORT RECOVERY**

Again user downtime is the penalty that must be paid as users have to terminate, partial transactions are deleted or completed, and then users are allowed to access the machine again. However, if strong locking is not in place, the transaction interaction that has been rolled out can inadvertently undo a completed call. The real solution to this dilemma is to have a "net change" rollback. This is currently unavailable, as a "net change" rollback requires a detailed and intimate knowledge of the application.

	HP 3000	<b>DB01/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	


## SOLUTIONS - HOW TO MINIMIZE RECOVERY HEADACHES

The benefit of such facilities as DBRECOV and Rollback recovery can be greatly enhanced if you implement the following safeguards.

1. Turn on logging - despite persistent misconceptions, logging does not significantly degrade the performance of your machine. If you are not logging you have precluded yourself from virtually all methods of recovery.
2. Use Begins and Ends - Without DBbegins and DBends, by definition no logical transactions exist. Therefore, database logical integrity is impossible to determine. The best that can be done is to provide for an audit trail of physical transactions.
3. Strong Locking - Some method of strong locking should be implemented. Without strong locking, a transaction can interact with another transaction before it has completed, thereby making the result of a rollback recovery questionable.
4. Turn on ILR - Turning on ILR will ensure that your database will always be physically intact.

Implementation of these four key points is crucial if you are to ensure database integrity and ease of recovery for your company. They can result in tremendous reductions in user downtime and the time is spent on recovery procedures. There is however, another alternative method of database recovery, that when implemented with the aforementioned safeguards, will render downtime due to the initiating of recovery, or the existence of logically corrupt databases due to program aborts, a thing of the past.



	HP 3000	DB01/6
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	


### **AN ALTERNATIVE - DYNAMIC "ROLLBACK"**

A facility that provides a dynamic rollback, will actually undo an aborted transaction as the abort occurs. This "real time removal" of an aborted transaction will result in your database always being logically intact. Without the existence of incomplete transactions in your database, it would also unnecessary to have to take the system down to initiate a cleanup.

Such a utility does exist, and is actually one product for the HP3000 from the Carolian Systems Research and Development group. Known as INTACT, this product provides these major capabilities which have been previously unaddressed and unavailable to HP3000 users.

Additional information on INTACT and other Carolian Products is available from:

Carolian Systems International Inc.  
3397 American Drive, Suite #5  
Mississauga, Ontario  
Canada  
L4V 1T8  
or call (416) 673-0400

	HP 3000	DB02/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

# USING DBchange TO IMPROVE YOUR DATABASE ADMINISTRATOR'S PRODUCTIVITY

by  
*Robert Ross*  
*Online Database Support Engineer*  
*Information Technology Group*  
*Cupertino, California*

## Introduction

Database structure definitions are very rigid, although with good reason. Nothing is more important to the user than the integrity of the data which took days or years to build and maintain. Unfortunately, the same rigidity can be very confining when it comes to modifying the database structure, usually to increase capacity, add new items, or increase performance. Image/3000 users have always had a way of accomplishing this through the use of DBUNLOAD and DBLOAD, a relatively cumbersome, slow, and inflexible method of changing the structure of a database, but one that does work for infrequent changes. Since many users needed faster and more powerful restructuring tools, several third-party software vendors produced utilities which made the process of modifying the structure of your database far less painful. Hewlett-Packard now offers its own database restructuring tool called "DBchange". This product, available for TurboImage databases and to be released on the Udelta2 MIT, will give database administrators and programmers the fast, disc-based database restructuring capability needed for large databases.

## What can DBchange do?

DBchange can make additions, deletions, and changes to a database. It can copy a database, review its structure, and print a schema from it. Passwords, data items, sets, paths, fields, and sort items may all be added or deleted. Item names, set names, the database name, set capacity, the blocking factor of sets, the device class on which sets reside, the set order within the schema, set names, user access to sets, field order within sets, passwords, user classes, item types, item names, paths, and sort items may all be changed. Sets, fields, paths, and items may be reviewed at any time. And any or all of the above may take place in one pass of DBchange, with the exception of copying. The following table will show the capabilities available in DBchange grouped by function.

**Table 1 - DBchange Functions**

ADD	CHANGE	DELETE
Passwords Data items Data sets Paths Data set fields Sort items	Passwords Data items Data sets Paths Data set field order Sort items Data set and data item access Data item attributes Primary paths Data item schema order Data set schema order Data set capacity Data set blocking factor Data set device class Data base name Data set and item names	Passwords Data items Data sets Paths Data set fields Sort items
REVIEW	COPY	PRINT
Data sets Data items Paths Data set fields	Data bases	Schemas

**Multiple Changes in a Single Pass**

DBchange's single-pass, multiple-change capability gives users the option of how many changes they want to make and when they want to make them. It also frees up the database completely for normal use while the changes are being entered and stored. This allows database administrators to accumulate changes during normal working hours, then perform the actual structural modifications to the database during off hours when no one is using the database. This would also allow a team of programmers, all working on different sections of the database for their application, to submit changes during one day and come back the next day with the database modified as requested, without interfering with other programmers.

### Modifying Item Attributes

Another aspect of DBchange that should be appealing to database/application programming and development teams is the ability to modify the attributes of any item. How many times have you wished that you had made a name or address field a little longer, or allowed for 99,999,999 orders instead of only 999,999? DBchange will make these and other changes to your items, without losing any of the data that exists already in the fields, even if the item is being used as a search or sort item.

### Type Conversions

DBchange supports type conversion for all data item types defined in the *TurboIMAGE Reference Manual* with the exception of I4 and J4. The following chart shows data item type conversions supported by DBchange. An x in the appropriate box indicates that you can convert the current item type to the new item type.

Table 2 - Supported Item Type Conversions

TYPE	I1,I2	J1,J2	K1	P	R2,R4	U	X	Z	←NEW ITEM TYPE
I1,I2	x	x	x	x	x	x	x	x	
J1,J2	x	x	x	x	x	x	x	x	
K1	x	x	x	x	x	x	x	x	
P	x	x	x	x		x	x	x	
R2,R4	x	x	x		x				
U						x	x		
X						x	x		
Z	x	x	x	x		x	x	x	

↑  
CURRENT ITEM TYPE

## How does DBchange work?

The product, "DBchange", consists of four files - two programs, a message catalog, and a forms file. A few changes to the Turboimage message catalog were also made, which is why DBchange should not be used on any MIT before UBdelta2. The user interface consists only of the two programs, called "DBchange" and "DBalter".

### Running the DBchange Program

DBchange is an interactive program designed to accumulate information from the user regarding modifications to the database. It is a VPLUS application which uses forms, and therefore cannot be run on any terminal that does not support block mode. It uses an inverted tree structure of screens, based on which function key the user presses, to display help information, review information, or collect changes from the user to store in a "change" file. A map of these screens (without the help screens showing) might give the user a better visual idea of his or her location within the DBchange program.

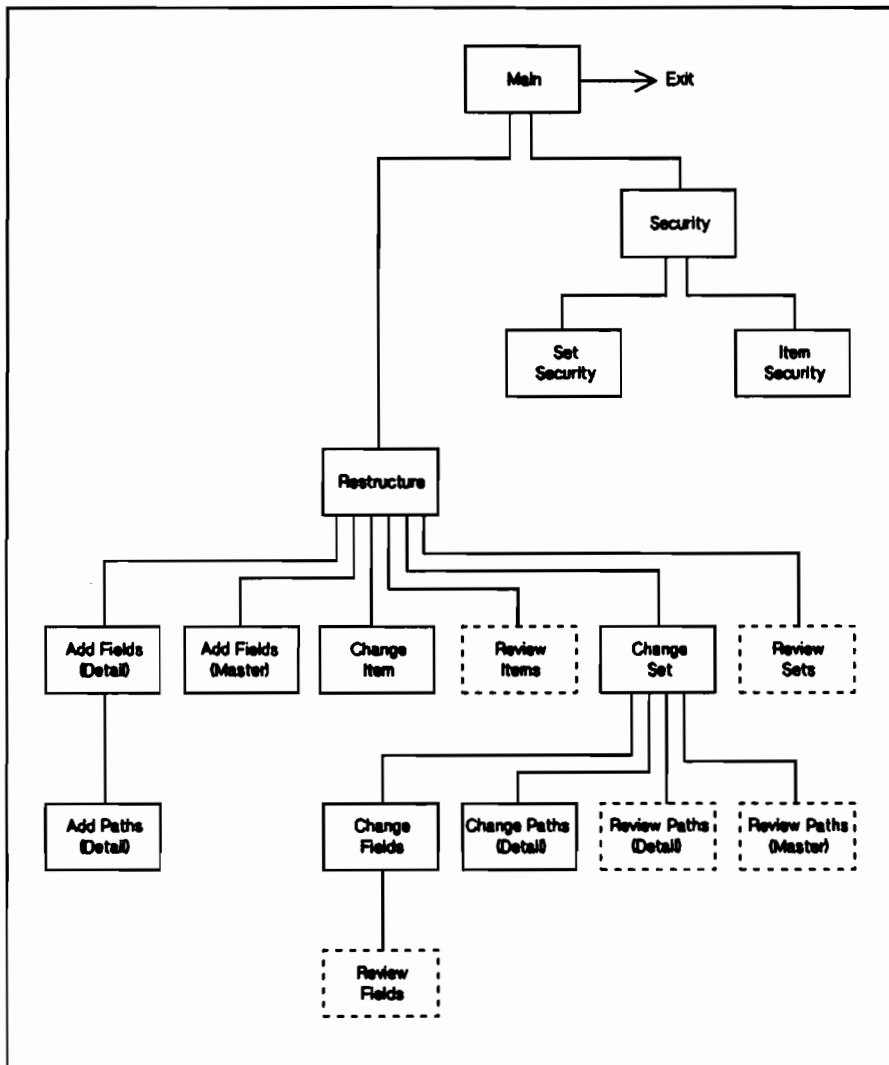



Figure 1 - DBchange screens

### Main Menu Options

After entering :RUN DBCHANGE.PUB.SYS, the user is presented with a main menu screen. Function keys 1-8 will change the name of the database, copy the database, modify its security, restructure its definition, print a current schema, perform the actual physical modifications to the database, display help information, or exit from the program. Since DBchange opens the database in Mode 5, others users can continue work as usual. Until the changes stored in the "change" file are physically applied to the database, nothing will change from the users standpoint. However, when reviewing items, sets, fields, paths, or printing a schema, DBchange considers all of these changes stored in the "change" file to be part of the existing structural definition of the database and will display or

	HP 3000	<b>DB02/7</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

### Keeping Database Copies Secure

In order to maintain database security, copying a database is not something that any user can do, but a privilege that the database administrator grants to a user. The administrator uses DBchange to enter the users name, plus the group and account that the database will be copied into. This is stored in the "change" file, and only that user logged on to that group and account will be able to make a copy of the database. The user will run DBalter within his or her group and account, enter the database name, and a new copy of the database will be created there. The conceptual difference here is that the user is "pulling" the database into his or her group and account only if the database administrator grants permission for the copy, rather than the administrator "pushing" a copy of the database into a user's group and account.

### The Change File

When a database name is given to DBchange to work on, DBchange looks for an existing "change" file for that database. The change file will be the name of the database with the letters "CF" appended to it. It is a privileged MPE file, and considered to be part of the database itself. Entering Purge, Release, or Secure in DBUTIL will affect the change file also. If DBchange does not find a change file, one is created. If it already exists, the user is given the option of purging it, purging it and creating another, or keeping it and adding more changes to it. The change file contains seven different types of records, four of which duplicate all of the information in the database root file, and three of which are changes to items, sets, or fields. All of these records are allocated at creation time, so any change records not used by the administrator when making modifications to the database are left blank. After DBalter completes successfully, the change file is purged unless DBalter was run with the correct parameter.

## How can you use DBchange productively?

The number of changes that can be executed during one pass of DBchange is almost infinitely varied, but with this freedom comes a responsibility to think about how each change will affect the performance of the restructuring pass of DBalter. Some of the less time-consuming transformations possible are any security alterations, changing the capacity of detail sets, adding new detail or standalone master sets, deleting sort items, changing the base name, etc. Some of the more time-consuming operations include reblocking sets, adding new paths, changing the capacity of master sets, deleting search items, etc. However, there are many modifications that might seem to be quick, but will end up taking an unexpectedly long time.

### Easy Changes That Take a Long Time

For instance, let's take a simple database that has a master set with paths to three detail sets, and imagine moving detail set number one to the end of the schema for some reason. This appears to be a fairly innocuous change, but it actually entails rewriting every record in the master dataset. The reason is as follows - if the master set is #1 and the three details are sets #2, #3, and #4, then within each master record there are pointers to all three detail sets ordered like this 2-3-4. After moving the first detail to the end, the pointers in the master are still in 2-3-4 order, but DBalter is about to change the detail set order to 3-4-2. Since DBSCHEMA always orders paths numerically, this 3-4-2 order will be switched around, forcing all of the master records to be read and then rewritten so the pointers can be changed from 2-3-4 to 3-4-2, a time-consuming process that is created by a seemingly quick and innocent change to the database. The twin brother of this problem comes from moving fields around in a dataset if they are search items.

### Reformatting Modified Data Types

Another request that will increase the time it takes for DBalter to restructure a database is item attribute changes. If any items are changed in a set, DBalter will have to read in a dataset record, deblock it into individual media records, format the old data types for each record into a buffer containing the new data types, change any pointers, if necessary, then write the block back out to disc. Since this is all done in memory, it's fairly quick, but it still slows things down somewhat.

### Are All of These Changes Necessary?

These caveats apply any time multiple changes are being considered for a database. If you as a database administrator are trying to decide whether or not to run a pass of DBalter during lunch time, and the main purpose of the change is simply to increase the capacity of a detail set, but someone wants to slip in the addition of a path or a blocking factor change, think about how long these additional modifications will take. You might want to do the capacity change right away, but put off the others until a weekend or holiday when more time is available.

### How can you use DBalter productively?

DBalter can only act upon the instructions stored for it by DBchange in the change file. DBalter runs with exclusive access to the database, and can therefore use output deferred mode when writing records, increasing throughput. It also uses large block reads and writes when it can. However, other than reducing the amount of work DBalter has to do by modifying fewer things, there is very little a user can do to speed things up once the restructure has begun. However, if DBalter is being run in batch mode, there are some parameters that can be passed which will be helpful in certain circumstances. The last four (low order) bits of the parameter each have a specific function, and they may be used in any combination. Their descriptions are as follows:

Parm	Value	Bit	Description
	1	15	Do not purge the change file after successful completion.
	2	14	Root file has inconsistency. If correctable, continue?
	4	13	DBSTORE has not been done on this database, continue?
	8	12	Not enough disc space for all temporary files, continue?


Bit 15 is simply a directive to DBalter to leave the change file alone when finished. Bits 12, 13, and 14 represent answers to questions that DBalter would normally ask when running in session mode. A value of "1" indicates a 'yes' answer to the question, a value of "0" indicates a 'no'. These parameter values are used only if the question is asked. Otherwise they're ignored. Since these parameters may be used in combination, any parm value from 0 (or none at all) to 15 is valid.

### What about database security?

DBchange was designed with the security needed by a database administrator in mind. It can only be run when the user is logged on as creator of the database in the group and account where the database resides. Simply knowing the database name or maintenance word will not allow other users to use DBchange to modify anything. Using standard MPE security, the administrator maintains complete control over all database modifications or copies.

#### Applying Changes to Multiple Databases

Unfortunately, sometimes this security can be confining, particularly in a development environment. What if a user enters a large number of changes for a database, and the next group over decides they want to do the same thing to their identically structured database? They could type in all of the changes again for their database. Or, if the first user runs DBalter with PARM=1, the change file will still be there when DBalter is finished. Can the second group just copy the change file to their group and account, then apply it to their database? Well, that depends. Remember that the change file contains all of the same information about the database that the root file contains. When DBalter is creating a new schema, it uses the tables stored in the change file. So if the second database has the same name as the first, it might work out fine, especially if all of the items, sets, and fields are defined the same way. However, the tables in the change file also contain item, set, and field security information, plus the capacities of all of the datasets. Again, if the security and set capacity of the

	HP 3000	DB02/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

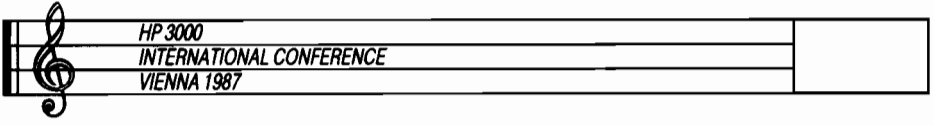
second database matches the first, this method will work correctly. But if they're not, the change file can still be used, but the second group will have to run DBchange, and enter change records for any security or capacity on their database that isn't the same as first database, where the change file was created.

## Advantages over previous methods


Prior to DBchange, there were three major methods of restructuring a database. First came DBUNLOAD and DBLOAD, which use tape or a serial disc for their storage medium. The Dictionary Utilities, DICTDBU and DICTDBL, which use disc storage, and the various third-party programs, all of which, I think, use disc as their storage medium, were introduced later. DBchange joins the major third-party programs in being a fast, disc-based, and comprehensive database restructuring tool.

Database integrity is of utmost importance when considering a restructuring utility. DBchange has been an extremely reliable product during pre-release testing, and continues to be. DBchange is easy to use because of a consistent, screen-oriented user interface that provides help screens at any point while entering changes. There aren't many files to install, and no changes to the system library. The installation process consists of copying the files from a tape into PUB.SYS, and the program is then ready to run. Administrators, programmers, and others can enter as many changes as desired while users continue their activities on the database. There are no limitations on the combination of changes that can be performed in a single pass using DBchange, and then the physical changes can be implemented in a job stream during the night or on weekends.





HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

	HP 3000	DB03/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Relational Access to IMAGE Data Bases

**Dr. Wolfgang Matt**

**Industrieanlagen-Betriebsgesellschaft mbH**

**Einsteinstraße 20, D-8012 Ottobrunn**

**West Germany**

### Introduction

Today everybody talks about Relational Data Bases. Even some HP 3000 users feel that IMAGE might be old fashioned and speculate about switching to SOL. A relational data base however is a theoretical concept. It says how one should look at data and what kind of operation should be performed on data. It does not say, how a data base should be implemented. In this paper it will be shown that IMAGE is not such a bad implementation of this concept as some people think.


### Requirements for a Relational Data Base

- a) The user's view of the data should be separated from the hardware realization.
- b) All data should be organised in flat tables.
- c) Each column has a unique name. All data in a column are of the same type.
- d) Each row has a unique key.
- e) Row and columns can be viewed in arbitrary order. The applications should work independent of the order.
- f) Different classes of users see different amounts of data. It should be possible to extract a subset of columns for some users, and join several tables for others.
- g) A data base should easily be changed.

### How IMAGE fulfils these requirements

- a) An IMAGE data base is largely independent from the hardware realization. The user only sees the data base name, not the physical realization in root file, ILR-file and date files. He opens only the data base, stores only the data base etc. IMAGE talks about entries which are not identical to logical or physical records of MPE-Files.
- b) IMAGE data sets can be viewed as flat tables. An IMAGE data base is usually described as a network. But it can as well be described as flat tables. You may have a Customer master and an Order detail which can be viewed as two tables both having the Customer Number as a key. The existence of physical pointers from master to detail and within a detail are details of the implementations.
- c) IMAGE data sets consist of items, each having a unique name, and the item values are all of the same type. An item may consist of subitems, but their number is always fixed, so the subitem number can be considered as an appendix to the item's name.
- d) Only master data sets have a unique key. In order to provide a unique key for all data sets, it is necessary to form a key combining several fields. This is not possible with IMAGE. But we market a software which allows multiple field keys. This software avoids long chained read.
- e) When accessing an IMAGE data set an item list should be used. The items are returned in the requested order independently of the physical structure. When the data base structure is modified, e.g. new columns added, applications run unchanged. Applications are generally independent of the physical order. IMAGE has the nice feature to store unsorted chains chronologically. You should not rely on this (this feature will go with TURBO WINDOW), but use a time stamp instead. The software we market allows sorted retrieval without sorted chains.
- f) Using different IMAGE passwords, different users can have different views of the data. Generally users only see subsets of the columns and tables defined in the data base. From the items available to a user a further subset can be formed using a selective item list.

Joining of several tables to a new table is not supported by the IMAGE intrinsic. We shall market an extension to IMAGE which will allow logical relations to be defined without unloading the data base, and will extend the DBFIND and DBGET procedures to allow arguments and lists parameters to include items from several data sets.

	HP 3000	<b>DB03/3</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

g) Changing the IMAGE data base structure is not an easy task. Up to now there was no HP supported software for structural changes, but there are several third party products available which do this job. Adding a column still requires a large amount of data to be moved physically, but there is no necessity to unload the whole data base. For adding and deleting keys, we market a software which performs this task very fast, since the data is not moved. The keys are stored in a separate data set of the data base, and are covered by logging, recovery, and DBSTORE. The keys can be used in DBFIND and DBGET calls and are maintained automatically by DBPUT, DBDELETE and DBUPDATE. They allow access by generic keys and match code, features not provided by standard IMAGE.



**Information Retrieval using Relational Access**

In the following a software product will be described that allows relational access to IMAGE data base. From the user's point of view, IMAGE data bases are a collection of several flat tables. A subset of these tables the users can access forms the input. The program joins the tables into a single table extracting a subset of the columns and a subset of the rows.


The program has been designed to be very user friendly. The user selects the input tables (data sets) and the columns (items) by simply marking them in VPLUS forms. Then he marks the items to be used for selection. These definitions are stored in a data base to be executed later.

When executing the enquiry the user enters the selection values. Either exact or approximate selection values can be entered. Approximate values may contain "@" for any number of any characters, or "?" for any single character in the given position, or ">" or "<" for bigger or less relations. The software finds the fastest way for executing the enquiry. Whenever possible, multi field access or generic access is used to avoid long chained or serial reads.

The resulting table can be viewed at the terminal, printed or stored on disc. After downloading, the table generated may be accessed directly by Lotus or DBASE applications.


**Conclusions**

IMAGE can not only be viewed as an implementation of a network structure, it can also be viewed as an implementation of the relational model. IMAGE allows the implementation of very efficient standard applications like order processing or financial accounting. Data collected in these systems can be accessed very easily by the end user using a relational view of the data base.

	<i>HP 3000</i>	<b>DB03/4</b>
	<i>INTERNATIONAL CONFERENCE</i>	
	<i>VIENNA 1987</i>	

**Biography**

Wolfgang Matt holds a PhD in physics. Since 1977 he works with IABG, a company with 1700 employees near Munich. He is head of a group of scientists, consulting HP 3000 users and developing individual software for them. He is the author of SI-IMAGE and ENQUIRE, products for index sequential and relational access to IMAGE data bases.

	HP 3000	<b>DB04/1</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## "@", "\*" and Other IMAGE Lists

Fred White  
Adager  
Apartado 248  
Antigua  
Guatemala

### Introduction

Each call to DBGET is a request for IMAGE to read a specific physical record of a specific dataset of a specific database and to extract those fields identified by the item names or numbers of the LIST parameter, concatenating them in the same left-to-right order as they are referenced in the LIST, and to return the result as a logical record in the user's BUFFER.

This mapping of a physical record to a logical record is done under control of a table of field numbers rather than a table of item names or numbers.

The term "LIST processing" is used in this paper to refer to the techniques employed by DBGET in transforming a LIST into such a table of field numbers.

This paper discusses DBGET's methods of processing:

- Item Name LISTS
- Numeric LISTS
- "@" and "\*" LISTS
- "0" and "," LISTS

It includes a brief description of the ways in which the LIST processing of DBPUT and DBUPDATE differ from that of DBGET and a comment on TurboIMAGE differences.


### Special Notes

The first word of a Numeric LIST is a count N (<128) of the number of item numbers contained in the remaining words of the LIST array. All other LISTS consist of ASCII character strings terminated by blanks or semicolons and left-justified on a word boundary. Since the leading character can not be a binary zero, the "value" of the first word of these LISTS always exceeds 255.

At DBOPEN time, the user is assigned an access class determined by the password supplied to DBOPEN. This access class provides the user with an access mode to each dataset of the database:

UNCONDITIONAL . . . this means that the user's read (or write) access to all of the fields of the dataset is unrestricted. This access is granted\* only when the user's access class is included in the dataset's "write class list".

- \* Actually, DBOPEN denies the user UNCONDITIONAL access to all datasets if the database is DBOPENed in mode 2 or 6. This is a documented "feature" which users are forced to live with. For some databases it is possible for a user to DBOPEN a database in mode 2 or 6 and be unable to read (or write) fields of a dataset which would be readable (or writeable) if the DBOPEN mode were 1, 3, 4, 5, 7 or 8. DBOPEN could have avoided this inconsistency by treating all modes the same with respect to dataset access.

	HP 3000	<b>DB04/2</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

**CONDITIONAL** ... this means that the user's read (or write) access to each field of the dataset depends on the access class being included in the read (or write) class list of the data item defining the field. This access is granted when the user's access class is included in the dataset's "read class list" but is not included in its "write class list".

**NONE** ... this means that the user has no read (or write) access to any of the fields of the dataset. This occurs whenever the user's access class is not included in the dataset's "read class list".

## Preliminaries

DBGET makes use of 3 tables residing in the Data Base Control Block (DBC) to create a Field Number Table which controls the subsequent mapping of a physical record to a logical record:

### 1. The Item Table (1 per database)

All item names reside in this table in the same order as their occurrence in the ITEMS part of the defining schema. The first entry is referred to as item 1, the second as item 2, and so forth.

### 2. The Dataset Item Table (1 per dataset)

This is a byte array of length  $N+1$  bytes, where  $N$  is the number of fields in the records of the dataset. Each byte except the last (which is zero) contains the item number (between 1 and 255) of the item in the schema which was used to define the corresponding field of the dataset. These item numbers are in the same order as their occurrence in the ENTRY part of the schema for this dataset. The first corresponds to field 1, the second to field 2, and so forth.

### 3. The Item Security Table (1 per database)

Each word contains information controlling the user's read and write access to the corresponding data item. The first corresponds to item 1, the second to item 2, and so forth.

Each dataset has its own Field Number Table which is  $N+1$  bytes long where  $N$  represents the number of fields in the dataset. Each Field Number Table is initially binary zeroes. Each Field Number Table also has a *full record flag* (a bit) associated with it. This flag is initially zero (OFF). It is turned ON (set to one) whenever a "@" LIST is processed and is turned OFF (set to zero) whenever a LIST other than "@" or "\*" is processed. When ON, IMAGE maps the entire record to or from the user's buffer as if it were a single field. When OFF, the mapping is done field-by-field under the control of the Field Number Table.

## LIST Pre-processing

DBGET checks the first word of the LIST parameter. If its value  $N$  is less than 128, the LIST is numeric of length  $N+1$  words. Otherwise, the LIST parameter is scanned for a terminating blank or semicolon. The position of this terminator relative to the start of the string determines the word length of the LIST.

DBGET copies the entire LIST (including the terminator, if a string) into the TRLR area of the DBCB. The terminating blank, if present, is replaced with a semicolon. The remainder of the LIST processing takes place on this copy.

DBGET identifies the specific type of LIST it is dealing with by inspecting the first word of this copy. The possibilities are identified in the following order:

1st word	LIST Type
"@"	a <i>full record</i> LIST
"**"	a "same as last time" LIST
< 128	a Numeric LIST
"0"	a Null LIST
","	a Null LIST
other	an Item Name LIST

### Processing an Item Name LIST

An Item Name LIST is the most data-independent form of a LIST. Naturally, in keeping with the "No Free Lunch" law, it requires the most time to process.

DBGET initializes the output Field Number Table to zeroes and sets the *full record flag* OFF.

It then processes each name in the left-to-right order of its occurrence in the LIST:

1. It extracts the name left-justifying it in an 8-word (16-character) array, (padded with trailing blanks, if needed). It performs a table lookup of this name in the Item Table yielding an integer I (an item number) determined by the position of the matching entry within the table.
2. It performs a table lookup of I in the Dataset Item Table yielding an integer F (a field number) determined by the position of the matching item number within the table.
3. If the user has only **CONDITIONAL** access to the dataset, DBGET checks the Item Security Table to verify that item I is read-accessible.
4. It verifies that the field number F is not yet a member of the Field Number Table and then replaces the first zero in the table with F.

The first of these steps takes an order of magnitude more time to perform than the other three combined and is the major reason why Item Name LIST processing is so much slower than the others.

### Processing a Numeric LIST


DBGET initializes the output Field Number Table to zeroes and sets the *full record flag* OFF.

It then processes each item number I in the left-to-right order of its occurrence in the LIST:

1. It performs a table lookup of I in the Dataset Item Table yielding an integer F (a field number) determined by the position of the matching item number within the table.
2. If the user has only **CONDITIONAL** access to the dataset, DBGET checks the Item Security Table to verify that item I is read-accessible.
3. It verifies that the field number F is not yet a member of the Field Number Table and then replaces the first zero in the table with F.

This LIST processing is identical to the last 3 steps of Item Name LIST processing described earlier. Omission of the the first step of Item Name LIST processing makes Numeric LIST processing an order of magnitude faster than the processing of an equivalent Item Name LIST.



	HP 3000	<b>DB04/4</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

### Processing an "\*" LIST

No other LIST requires less time to process. All processing is complete upon recognition (i.e., the Field Number Table is left in its previous state).

### Processing an "@" LIST

If the *full record flag* for this dataset is ON, all processing is complete. In this case, the "@" LIST is as fast to process as the "\*" LIST.

Otherwise, the LIST processing depends on the user's access to the dataset:

With UNCONDITIONAL access, the Field Number Table is set to the integer values 1, 2, . . . , N (where N is the number of fields in the physical records of the dataset) and the *full record flag* is set ON.

With CONDITIONAL access, the LIST in the DBCB is initialized as a Numeric LIST with item numbers identical to those in the Dataset Item Table. The LIST is then processed and the *full record flag* is set ON.

### Processing "0" and "," LISTS

These are referred to as Null LISTS. DBGET zeroes the Field Number Table and sets the *full record flag* OFF.

Note: A Numeric LIST whose first element (number of items) is zero is another form of a Null LIST which happens to be processed as a Numeric LIST.

DBGET honors a Null LIST by returning a zero-length record to the user's buffer. This also occurs if an "\*" LIST is used in the initial access to a dataset. This is because each Field Number Table is initially all zeroes.

### Completing the DBGET

If the *full record flag* is not ON, DBGET must map the fields of the physical record, as determined by the Field Number Table, to a logical record.

DBGET uses the Field Number Table along with the Dataset Field Offset Table (not mentioned earlier) to construct this logical record in the TRLR area of the DBCB:

It initializes a logical length (LL) to zero.


It processes each field number F of the Field Number Table in the left-to-right order of its occurrence as follows:

The field length FL is calculated:

$$FL = \text{Offset}(F + 1) - \text{Offset}(F)$$

FL words are moved from Offset(F) of the physical record to TRLR(LL) and LL is incremented by FL.

The LL words of this logical record are then copied from the TRLR area to the user's BUFFER.

	HP 3000	<b>DB04/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Although this mapping is accomplished in an efficient manner, it does constitute additional overhead proportional to the number of fields referenced in the original LIST.

If the *full record flag* is ON, DBGET ignores the Field Number Table and simply copies the physical record directly into the user's buffer, bypassing the field-by-field mapping into the TRLR area and thus saving the CPU time consumed by this mapping.

### Performance Comments

If independence from database structure is a major consideration, your application should use an Item Name LIST. If this LIST is a constant, your application can avoid the CPU overhead associated with the processing of Item Name LISTs by using an "\*" LIST in all subsequent calls.

If you are accessing ALL of the fields of a dataset in the same order as they exist in the physical records and if independence from database structure is not critical and if speed is, your application should use a "@" LIST for the first LIST and an "\*" LIST for all subsequent calls.

The advantage of the "@" LIST is that the mapping based on the Field Number Table is bypassed since DBGET simply copies the physical record directly into the user's buffer in one step.

### LIST Processing Nuances of DBPUT

No LIST processing occurs unless the database was opened in mode 1, 3 or 4 and the user has UNCONDITIONAL access to the dataset.

DBPUT's LIST processing differs from DBGET's in that no field security check is made since the user has UNCONDITIONAL access to the dataset which, by definition, implies write access to all of its fields.

DBPUT verifies that all critical fields (search and sort fields) of the dataset are included in the LIST.

If the database was opened in mode 1, DBPUT must also verify that the calling process has a covering lock. This can be a database lock, a dataset lock or a predicate lock.

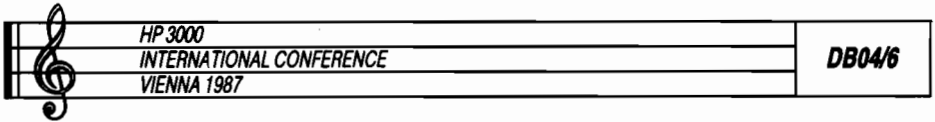
DBPUT's mapping is from logical record to physical record. Prior to this mapping, the physical record (in an IMAGE buffer) is set to binary zeroes. Consequently, any fields not included in the LIST will always be entered as zeroes.

### LIST Processing Nuances of DBUPDATE

No LIST processing occurs unless the database was opened in mode 1, 2, 3 or 4 and the user has access to the dataset.

The LIST processing of DBUPDATE differs from DBPUT in that the field security test is made if the dataset access is CONDITIONAL, which is always the case if the database was opened in mode 2. Also, DBUPDATE does not verify the presence of critical fields in the LIST.

If the database was opened in mode 1, and the user does not have a covering database or dataset lock, DBUPDATE must also verify that the caller has a predicate lock covering the new and the old values of the lock field.



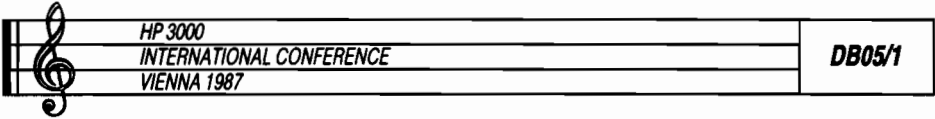
Like DBPUT, DBUPDATE performs a logical record to physical record mapping under control of the Field Number Table. Unlike DBPUT, the physical record being updated is not zeroed prior to this mapping.

DBUPDATE compares each new field value with the old one. If they differ, the field is verified not to be a critical field before its value is replaced.

### **Some Comments on TurboIMAGE**

IMAGE allows 255 data items per database. TurboIMAGE allows 1023 data items per database. To support this change, the TurboIMAGE Item Table was increased in size by about 8%. More importantly, each Dataset Item Table had to be changed from a byte array to a word array. This doubled the size of all of these tables which reside in the user's Data Base User Local Control Block (DBU). It also required that each table lookup performed on the Dataset Item Table be implemented in a program loop. The same lookup under IMAGE was performed with a single SCAN UNTIL instruction. Consequently, this table lookup is slower under TurboIMAGE than it was under IMAGE.

Since TurboIMAGE allows up to 255 fields per dataset, the first word of a Numeric List parameter must be less than 256. This is a trivial difference from IMAGE (which allows up to 127 fields per dataset) and is mentioned only for completeness.



HOW TO BUILD A DISTRIBUTED M.I.S. SYSTEM

Roger W. Lawson

Proactive Systems Ltd  
Central Court  
Orpington  
Kent BR6 0JA  
England  
Tel:0689-77933  
Telex:9413362

## INTRODUCTION

Management information systems (M.I.S) have historically been seen as centralised systems. The typical large company organisation structure is a hierarchy with a peak at head office. Therefore computer systems to support this structure have been concerned with the consolidation of locally collected data towards the centre and the distribution of information from the centre. In the past this often resulted in a naturally centralised approach to the provision of data processing resources. However this strategy is not only inefficient and inflexible but is also needlessly expensive. I will attempt to show how such systems can be built using networks of minicomputers (eg. HP3000s) and discuss the practical problems that have to be overcome.

## THE REQUIREMENTS

To support any complex data processing system or M.I.S. system that comprises multiple applications you need the following capabilities:

- The ability to share data between applications.
- The ability to pass transaction data from one application (or process) to another.
- The ability for users to access data from multiple data files easily and transparently.

Now if all the systems are running on the same computer, this is easy. However the centralised approach has many disadvantages. Some of these are:

- You are placing all your eggs in one basket. If the central computer stops then all application systems stop. Even a short interruption is very costly and the company as a whole may be vulnerable if a real disaster such as a fire occurs.
- The data communication costs are very high if the users are geographically dispersed. If the computer is located in Los Angeles but some of the users are in New York it is a very expensive solution.

- It is inflexible as an upgrade of processing power may only be possible in steps (also there tend to be upper limits on such equipment as is so with the power of the HP3000 computer range).

- Groschs Law may no longer apply. Multiple small computers can give you cheaper power than one large one.

#### THE PROBLEMS OF DISTRIBUTED SYSTEMS

If you spread applications over multiple computers then to meet the requirements mentioned above is much more difficult. Let's take each in turn:

- You need to be able to share data between computers that are geographically remote (although probably linked by a communication system such as DS/3000 even though this may be subject to frequent failure or interruption). For example you may be running a sales order processing system on one computer and a production/inventory system on another computer - they both need to share (and update) the same stock availability data. Although DS/3000 allows access to remote data bases it provides no facilities to logically link two or more data bases on separate computers.

- Moving transaction data from one system to another is much more difficult in a distributed system. An example is where a sales transaction needs to reduce a stock balance which is held on a local computer plus create an accounts ledger entry on another central computer. DS/3000 provides the facility to easily copy a file in "batch" mode but building a real time processing network with automatic recovery in case of a failure is another matter.

- User access to the data for both read and update purposes introduces technical problems. For example on an HP3000 if you have more than a few remote DS sessions then performance tends to be very poor. Also if you have a reasonably complex network (ie. more than 2 computers) and you have lots of remote data base access then you are very vulnerable to failure of any one node or any communication link. Obviously the more computers you have then the higher the risk you run of one being out of action.

## OTHER REQUIREMENTS

Other common requirements that are not covered easily are:

- Semi-static data on one computer needs to be reflected on other computers in the network. For example a price list that is maintained centrally needs to be distributed around the network. Note that in this case because the data does not change very frequently and is not large in volume it is more cost effective for each computer to have a local copy. The trade off is the cost of disc space against the communication and performance cost of remote data base access.

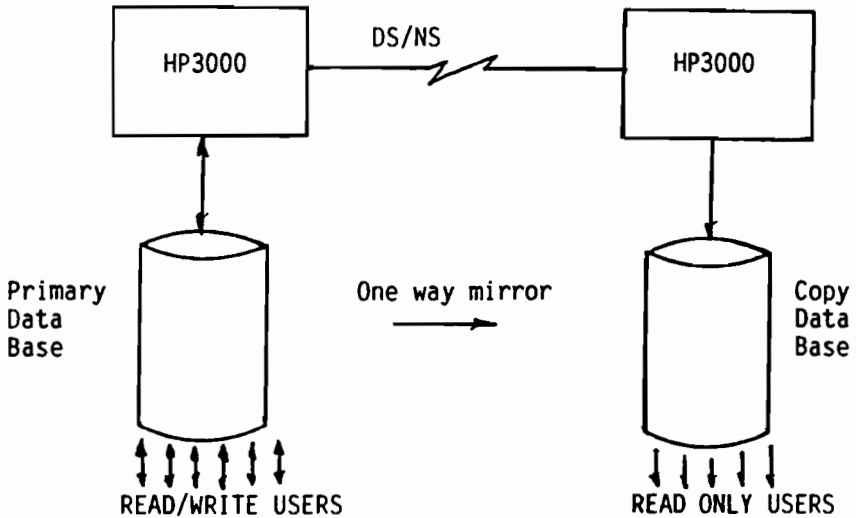
The "local copy" approach also makes each local system less vulnerable to network failure. Now you could do this by passing transaction data around the network and having some local processing code on each system, but all you really need is some system software which can be instructed with a command that says "keep the price data sets on computers 2, 3, 4 etc. automatically the same as the price data set on computer 1".

- Data consolidation is often required. For example sales transactions that are processed by each local computer need to be collected and merged in real time into a single data set on a central computer.

## A SOLUTION

Now a couple of years ago my company came up with a solution to the above problems which is a software product called BACKCHAT. The original germination of the product stemmed from both my and a colleagues experience in running multiple HP3000s as users (for example the company I worked for as DPM had over a dozen linked machines). Originally BACKCHAT was aimed solely at providing a real time copy of an IMAGE data base on a second computer by "mirroring" the data base. This application is very similar in purpose and mode of operation to HPS SILHOUETTE product (BACKCHAT is an alternative to that but with many more facilities). This mode of operation is represented in Figure A below.

Figure A



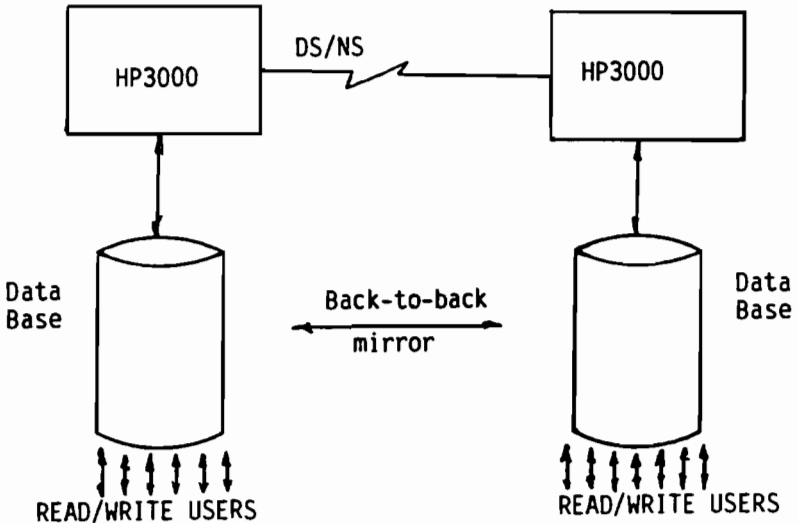
In this case the copy data base can be read by users on the secondary HP3000 but cannot be updated. This system provides:

1. A disaster protection system (users on the primary system can be switched to the secondary if the primary fails).
2. Load spreading as reporting/enquiry users can be off loaded to the second computer.
3. Concurrent back-up and 24-hour availability as tape stores can be done on the secondary without stopping access to the data base on the primary.

After releasing the product we were approached by a potential client who was not only interested in the disaster protection capability but who also had a number of other requirements to enable him to build a distributed system spanning England, France, Germany and Australia. One of his needs was to be able to logically share a data base (or data set) with updating on both systems. This is represented in Figure B.



Figure B



So we enhanced the product to provide this functionality (like many advances in the state of the art of computing the development grew out of close interaction between a software company and a user). With the new capabilities we can supply all the distributed data base requirements mentioned above.

#### TECHNICAL DESCRIPTION

BACKCHAT works by using the IMAGE logging system to pick up data base changes, passes the transactions to a remote processor and applies them to the remote data base. With concurrent updating as shown in Figure B there are effectively two of these processes in operation (one in each direction) so that the data bases can be viewed as being mirrored "back-to-back". There is special logic incorporated to stop transactions echoing backwards and forwards for ever. There is also special logic to cope with record relocations (record numbers of IMAGE records in detail data sets changing from one data base to another).

There is a lot of control and configuration logic associated with controlling:

- Multiple data bases concurrently.
- Multiple remote connections.
- Selective parts of data base (eg. data sets).
- Restart and recovery from failure.
- Simple operator control from one location.

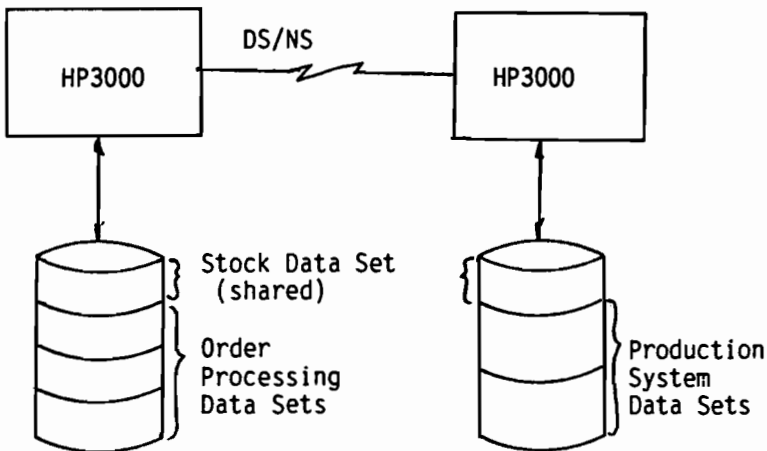
Note that BACKCHAT uses the IMAGE logging system for two reasons:


1. It is the most efficient way of picking up data base changes, i.e. has least performance impact.
2. It contains logical transaction definitions which may need to be used in recovery situations.

Incidentally BACKCHAT does not use privileged mode.

With this kind of software one can easily "share" information across a network without special programming. For example with a simple configuration dialogue it is possible to set up sharing of stock information as in Figure C below.

Figure C



	HP 3000	DB05/8
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

As you can see the data bases do not need to be similar except for the data set that is to be shared.

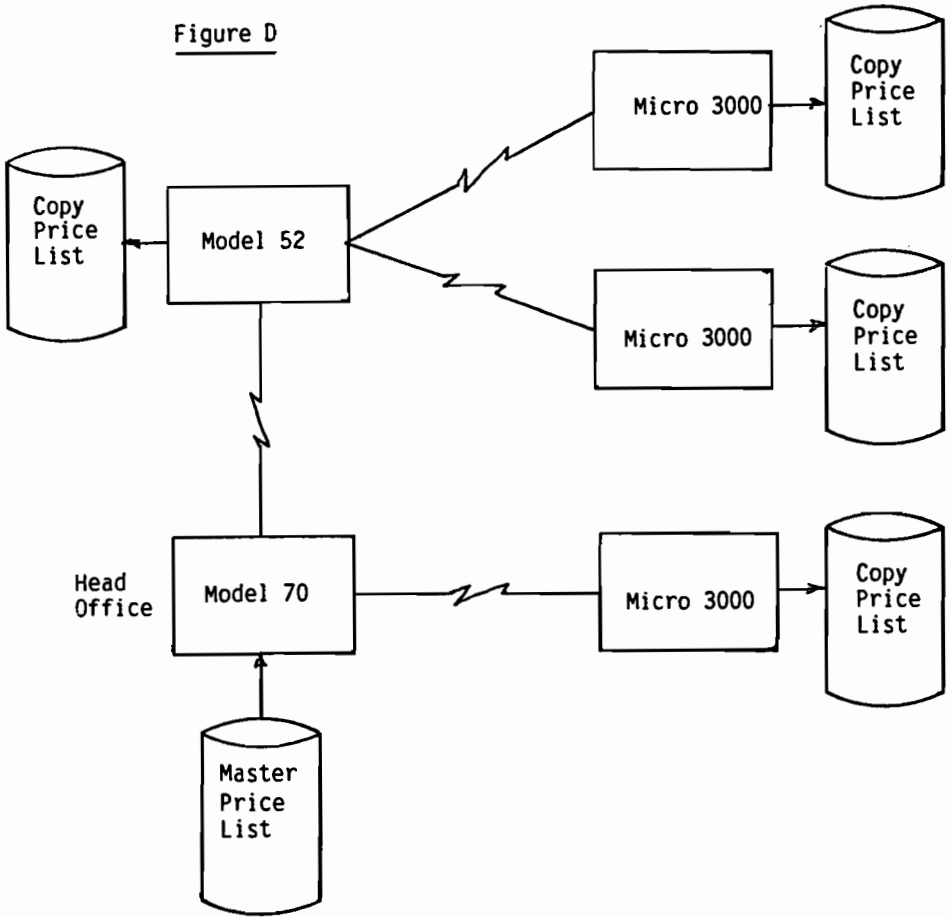
#### CONCURRENT UPDATING

Now the obvious question with this design is "what limitations are there on concurrent updating". Firstly if the data sets within a data base are only updated on one system then no potential conflicts arise. Also the software can handle updates to the same data set so long as the particular records being updated are different. However because it is not practical to impose a global lock (which would reduce the tolerance to network failure in any case) the concurrent updating of the same record has to be considered with care. For example if a record was deleted on one system at the same time as it was updated on the other then when the update arrived on the first system (which may be some time later depending on configuration etc) the record would have disappeared. However with suitable application design it is possible to avoid these problems (eg. flag the record for later deletion in this example). Although it may not be possible to take an existing application running on a single computer and chop it in two without application changes, there are effectively no significant limits on what you can achieve.

#### MORE EXAMPLES

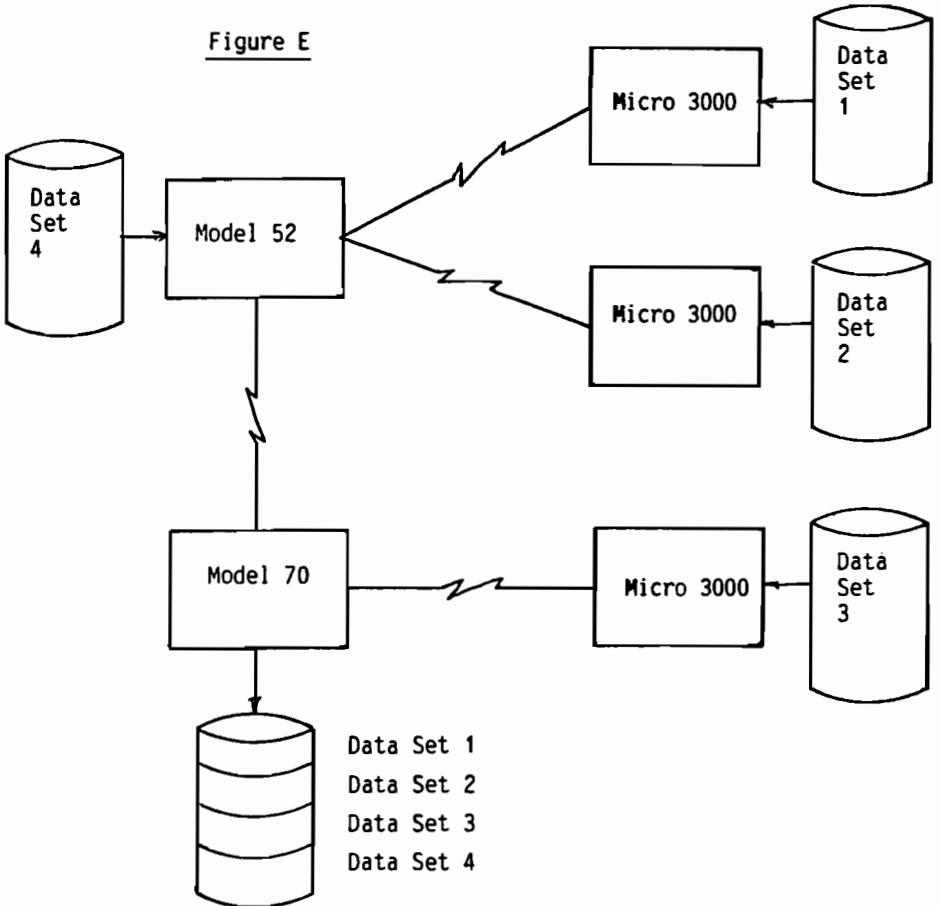
Figures D and E below show how easy it is to provide the shared reference information and data consolidation facilities.

Figure D



In this example head office maintain the master price list on their computer and BACKCHAT automatically maintains copies on the other computers (the intermediate copy on the Model 52 could be dispensed with if not required).

Figure E



In this example branch sales are automatically passed from each local system as they are collected to the central system. They are then automatically consolidated into a single data set on the central system. Note that BACKCHAT also contains utilities to help set up the merged data set initially. In this example transactions are effectively transmitted through the network for further processing at another location.

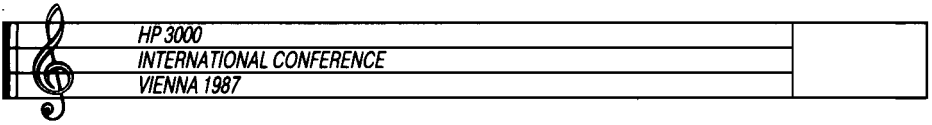
## TOLERANCE TO FAILURE

Because BACKCHAT processes and manipulates log records in files on both local and remote systems these files effectively act as buffers in the network. This means that the separate computer processors can operate asynchronously and, if a network failure occurs, transactions simply accumulate in the buffers until the relevant part of the network is restored (BACKCHAT can then automatically recover to the correct point). Whether it is communications line failure or computer failure, in neither case are other computers in the network significantly affected using this approach.


The software also contains such facilities as a roll back recovery module that is invoked in certain failure circumstances (it includes linked roll back of logical transactions that span multiple data bases by looking at the user PIN number to identify common usage).

## CONCLUSION

I hope I have shown how it is possible to build distributed data base solutions now on HP3000 equipment using IMAGE. With the approach described all the requirements of distributed data processing can be easily met. It is certainly practical to build real integrated networks and do the computer processing in the locations best suited to minimise costs and maximise user convenience.



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

	HP 3000	DB06/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## NORMALIZATION - THE PERFECT DATABASE?

Glen Kalina  
Information Technology Group  
Hewlett-Packard, Cupertino, CA., U.S.A.

### INTRODUCTION

As a relational database designer, you may be faced with a non-trivial task when creating a schema for your database. Common sense may suffice for very small databases, but what does one do when faced with creating large databases? Fortunately there are rules that the designer may choose to follow when creating the database schema. This paper will explore the advantages and disadvantages of the rules which make up Normalization Theory.

### NORMALIZATION THEORY

Normalization theory is composed of several normal forms. Normal forms are merely rules that may be used as guidelines in the creation of a relational database schema. Over the years many normal forms have been defined. E. F. Codd, considered the founder of relational database theory, was responsible for 1st, 2nd, and 3rd normal forms. First normal form is the most basic and each successive normal form is more rigorous than the previous. Since there are several normal forms, a relation is said to be in a certain normal form if it satisfies all the rules related to that normal form and all normal forms preceding it. Thus, for a relation to be in 3rd normal form, it must already be in 1st and 2nd normal forms. When it was discovered that 3rd normal form suffered from some deficiencies a stronger normal form was derived to replace it. This new normal form is called Boyce/Codd normal form. Later, R. Fagin defined 4th and 5th normal forms which normalize a relation even further. Later on, we will examine the advantages and disadvantages of first through fourth normal forms. As these rules are based upon theory, it is important to note that the database designer need not rely upon normal forms as absolute laws of relational database design. Normalization theory should be used only as a guideline in database design since there are often very good reasons for not fully normalizing a database. As the designer, it is up to you to determine which tradeoffs are attractive and how far your database should be normalized. The extent to which the database needs to be normalized



is application dependent, so keep in mind that normalization should be "taken with a grain of salt" when designing a schema.

## WHY NORMALIZE A DATABASE?

There are many different reasons why the designer would want to normalize a database. The major problems that normalization will rectify are those of redundancies and anomalies. As an example let us examine the following relation

manufacturer(name,address,part,cost)

Redundancies arise when a when information is needlessly repeated in a database. The potential for wasted space can be great in a large database environment. If the storage savings will be significant then it makes sense to normalize and remove redundancies. Besides wasted storage space, there is another reason why redundancies should be removed. There is the possibility that multiple copies of the same information will be altered in different ways so that they no longer agree. In our example above, the address of the manufacturer is repeated one time for each part that manufacturer supplies. By normalizing this relation, we can eliminate this redundancy.

The other major problem that normalizing will take care of is that of insert, delete, and update anomalies. Insert anomalies occur when we can't add a manufacturer's address because that manufacturer does not supply any parts at that time. Null values could be used as a workaround, but would present additional integrity problems for the application programmer. A delete anomaly is the inverse of an insert anomaly. If all the parts supplied by one manufacturer are deleted from a relation, then we'll lose track of the address associated with the part. The update anomaly is related to the redundancy problem we spoke of previously. Because of the redundancy of a certain field, it is possible to update the value of a field in one tuple while leaving the same value unchanged in another tuple. For example, we could update the address for a manufacturer in one tuple while leaving the old address unchanged in another tuple.

## RELEVANT DEFINITIONS

Before we proceed to examine normal forms, we need to define a few concepts relevant to normal forms and relations. By definition, every relation will have a key. A key is merely a set of fields, one or more, that will uniquely identify a tuple. To be more specific, a candidate key is a minimal set of fields that uniquely determine a tuple. We can choose one of these candidate keys to be our primary key. This primary key will consist of one or more fields and has the property that no two tuples may contain the same values in their respective key fields.

We now need to define functional dependencies. A field Y of a relation is functionally dependant on a field X of the same relation if for each value of X there is only one value of Y. In other words, a given value

of X must always occur with the same value of Y. To carry the definition further, for a field of a relation to be "fully" dependant on X of the same relation, Y cannot be functionally dependant on any proper subset of X.

## FIRST NORMAL FORM

For First Normal Form to be satisfied the following condition must be met. At every row and column position (a field) within the relation there can exist only one value. So all field values are atomic and no field contains repeating groups. Repeating groups are sets of data values that could exist in each field instead of one value. Relational databases by definition don't contain repeating groups, so all relations are in First Normal Form.

Since all relations are in First Normal Form, we have no choice as to whether we wish to normalize our database this far. But, as we've banished repeating groups in fields, we gain the advantage of simplicity in design and lose the headache of keeping track of multiple values programmatically.

## SECOND NORMAL FORM

A relation is in Second Normal Form if it is in First Normal Form and all the nonkey fields are fully dependant on the primary key. The goal is to eliminate non-full dependencies by removing all nonkey fields that are also dependant on a subset of the key. As an example, let us examine the following relation.

```
inventory(factory,part,cost,address)
```

The primary key in this relation is the composite key consisting of the factory and part fields. But this relation violates Second Normal Form because the address field is not fully dependant on the key. It is in fact dependant on a subset of the key, factory.

To satisfy Second Normal Form the relation should be replaced by the following two relations.

```
factory_address(factory,address)
```

```
inventory(factory,part,cost)
```

The field factory is now the primary key of the relation factory\_address and the fields factory and part form the primary key of the relation inventory. As we can see, all nonkey fields are now fully dependant on the primary key.

What have we gained by normalizing the initial relation into Second Normal Form? Since the address field was repeated once for every occurrence of a factory, we've eliminated redundancy. This has two effects. First, if the relation contains a large amount of tuples, we've saved alot of storage space.

Secondly, we've eliminated the problems arising from the insert, delete, and update anomalies we spoke of previously. We now have an address for each factory even if that factory has no parts available. Furthermore, if we delete all the parts a factory is responsible for, we still have an address for that factory. Finally, since the factory address is recorded in only one place, we avoid the update anomaly associated with redundancy.

But, we do pay a performance price for this normalization. It is important to remember that this price is wholly dependant on the size of the relations and the application accessing the database. Because the address of a factory is now in a separate relation, a join of the two relations must be done if we wish to see the address related to each part manufactured. As a rule, the greater the number of relations in a join, the greater the time spent by the software in retrieving the data from multiple relations.

### THIRD NORMAL FORM

A relation is in Third Normal Form if it is in Second Normal Form and every nonkey field is not transitively dependant on the primary key. Put more simply, a nonkey field must be fully dependant on the primary key but cannot be functionally dependant on any other nonkey field. Let us use the following relation as an example.

allocation(engineer,group,site)

The primary key in the relation is the engineer field. Assuming that an engineer only works for one group. If we further assume that each group is located at only one site, then it becomes apparent that the site field is functionally dependant on on the group site while also being functionally dependant on the key engineer field. This is what we mean by a transitive dependency. To satisfy Third Normal Form the allocation relation should be replaced by the following two relations.

allocation(engineer,group)

location(group,site)

The decomposition into Third Normal Form has benefitted us in several ways. The redundancy due to the site field being repeated for every engineer has been removed, saving space and avoiding the anomalies due to redundancy.

Again we pay a performance penalty. The performance loss is due to the join we must now perform to retrieve all the information previously located in one relation. As before, the performance penalty will be related to the size of our relations and and the number of joins.

#### FOURTH NORMAL FORM

A relation is in Fourth Normal Form if it is in Third Normal Form and it does not contain two or more multivalued dependencies. Multivalued dependencies can be many-to-many or many-to-one relationships. Following is an example of a relation with a two man-to-many relationships.

```
programmers(engineer,language,host_machines)
```

An engineer can have the ability to write programs using several languages on several machines. Thus we have two many-to-many relationships.

To satisfy Fourth Normal Form the relation should be replaced by the following two relations.

```
programmers(engineer,language)
```

```
machines(engineer,host_machine)
```

As can be seen, we have split the two multivalued relationships into two separate relations. The benefits aren't apparent until you try to design an application that will guarantee the integrity and consistency of the data. When there are several multivalued relationships in a relation the designer has to create a sophisticated relation scheme that can cover all the tuple combinations possible from many-to-many relationships. If insertions, deletions, and updates aren't done very carefully, data will become inconsistent. By normalizing a relation with multivalued dependencies it we remove the possibility of potential inconsistencies while simplifying the relation scheme and programming effort.

Once again we pay the same performance penalty as for the previous three normal forms.


#### CONCLUSION

After looking at four normal forms we can make the following observations. Using normalization we simplify a database by removing redundancies and the insert, delete, and update anomalies. By doing this we have made the database very efficient from a maintenance standpoint. The programming effort required has become simpler and smaller. Since certain consistencies and integreties are now inherent in the database design, an application can concentrate on simpler transactions without the hassle of of consistency checks.

The physical outcome of normalization is lots of small relations since we have decomposed complicated relations into an increased number of simpler relations. I have seen normalized databases that end up larger than the unnormalized versions but the converse is often true. If the redundant fields that have been removed are large, such as an address, then more often than not the normalized database will occupy less physical space than the unnormalized version.

It has been our experience that increased normalization will increase data access costs. By splitting fields into many relations we create a situation where joins must be carried out to retrieve data that was previously located in one relation. There is no question that joins can be expensive. I have at times seen a manyfold increase in retrieval time from a join of two relations, one with ten thousand tuples in it, when compared with the original ten thousand tuple relation. The retrieval cost will only rise with the amount of table joins so bear in mind exactly what simplicity in design is costing you.

In a nutshell, it is your choice as to how far you wish to normalize your database. You must choose a point on the normalization scale that is a best fit for your application. By experimentation with prototype databases you should be able to create the optimum compromise where the tradeoffs between simplicity, efficiency, and access costs are balanced out.

	HP 3000	<b>DB07/1</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

A Comparison of TurboIMAGE and HPSQL  
by Larry Kemp, HP Bellevue, WA

This paper is intended as a primer on HPSQL for current users of the IMAGE database management system on HP3000 computers. SQL, which is an acronym for Structured Query Language, is the new relational database management system for the HP3000 family. SQL was originally implemented on IBM mainframes, and has since been implemented on several other computer systems. SQL is an implementation of the original "System R" specification for relational databases. The ANSI committee has accepted SQL as the relational database model.

Users of TurboIMAGE will find that HPSQL provides considerably more flexibility than does TurboIMAGE. IMAGE has probably gained most of its popularity due to its ease-of-use and simplicity of design aspects. SQL should provide even more ease-of-use and simplicity.

IMAGE has gained popularity due to its good performance, predominantly to do with the ease with which the designer can take performance into account. For example, the IMAGE designer can effectively, easily, and accurately utilize blocking factors.

Another area where IMAGE excels is having a considerable knowledge and experience base. IMAGE is installed on all HP3000 computer systems, and IMAGE is the database management system used for most HP3000 applications. Therefore, there is considerable expertise available on good IMAGE design, both from HP and from a large number of third party consultants. The IMAGE handbook exemplifies the public knowledge base. There are a number of well known implementation (and optimization) techniques for IMAGE.

There is a knowledge base for SQL, and for the most part that knowledge focuses on high level design issues. There are well documented logical database design techniques that utilize relational database constructs, one example which is the normalization of databases to "third normal form".

The last, very positive trait of IMAGE has been its reliability. IMAGE databases rarely, if ever have integrity problems. And when some damage does happen, there are accurate, if not time consuming, recovery techniques. Since SQL is new, its reliability remains to be seen. SQL does have automated logging and rollback recovery, so SQL databases should not have integrity problems.

The remainder of this primer will focus on the usage and features of IMAGE and SQL on a sample database and problem. I will focus on data structure and design, query (data manipulation) language, program-and-data independence, security, and transaction management. I feel that these are the reasonings for databases.

### Structure.

IMAGE and HPSQL use different terms to describe database structure. IMAGE uses the term "sets" to describe logical groupings of like described data. A non-database user would call that construct a file, with a restriction that all of the records are of the same record-layout. An SQL user calls that construct a "table". The IMAGE user refers to repetitive occurrences in the set as "entries", while the non-database user refers to that construct as records.

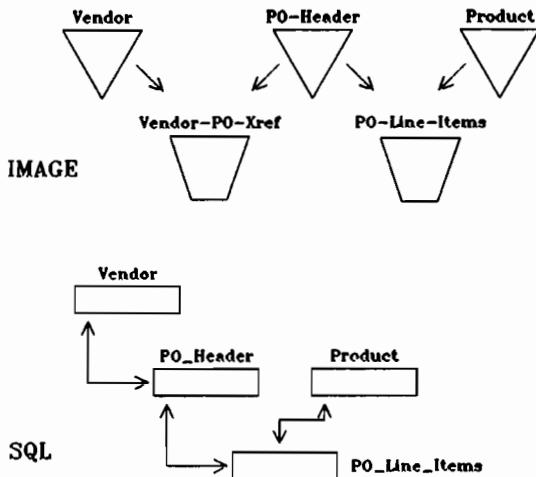
The SQL user refers to those constructs as "rows". And lastly, the IMAGE user refers to the individual components of an entry as "data items", where the non-database user refers to them as fields. The SQL user refers to "columns"

<u>Non-Database</u>	<u>IMAGE</u>	<u>SQL</u>
File	Set	Table
Record	Entry	Row
Field	Item	Column

IMAGE datasets are defined as one of master datasets, or detail datasets. Master datasets have unique keys and can be accessed by key or sequentially. Entries in a detail dataset are chronologically organized by common key. Entries can be accessed either sequentially, or along the chronological key path. Master datasets can be related to details, and in a logical sense, detail datasets can be related to masters. This results in the definition of IMAGE has an extended two level hierarchy.

SQL makes no distinction of master versus detail datasets. Any two tables can be related, allowing multi-level "Join" operations. And any table can be accessed either by key/path, or sequentially. Furthermore, a given table can have multiple keys, including keys which are formulated from several columns. Generic and approximate searches are allowed.

Here is an example implementation using the two database management systems:



The most noticeable difference between the two implementations is the lack of an artificial connecting dataset between Vendor and PO-header in the SQL database. Just as worthy, is that the SQL database implements the same functionality as the IMAGE version. SQL, as in IMAGE, has the ability to declare unique keys for both the Vendor and PO-header tables. Also, the PO-number index for the PO-line-items dataset can be declared "clustering", which



allows optimized physical placement along that index, in an analogous technique to the "primary path" for IMAGE detail datasets.

### Query Language.

Most users of IMAGE were introduced to IMAGE through QUERY. QUERY has a simple, English-like syntax that allows command driven access to IMAGE databases. Query is a good learning tool in that it is easy to learn, and allows exercising of most IMAGE functions. Once the novice has mastered QUERY, he/she next learns how to programmatically access IMAGE. This involves formatting subroutine calls to IMAGE. One record is accessed at a time, with one or two calls necessary to access each record.

SQL, like IMAGE, has an ad-hoc program for accessing databases. (The SQL program is called ISQL, where the I stands for Interactive.) Most SQL users learn SQL through ISQL, in an analogous manner to the IMAGE user with QUERY. But unlike the IMAGE, programmatic access to SQL is nearly identical to ISQL access. In other words, the user codes the same commands programmatically as he/she uses in ISQL. Consequently SQL is easy to learn.

Like Query, SQL allows conditional specifications of rows to be selected. And like Query, SQL uses that specification to determine the access method. The access method is determined by SQL, and not by the application program.

SQL implements the query language in COBOL and PASCAL by using pre-processors. These pre-processors translate the high-level query commands into the appropriate subroutine calls. The only difference between the interactive commands and the programmatic SQL commands are the specification for where the resulting data resides. (Programmatically, the INTO clause is specified which says where in the program to store the result of a command.)

Here are sample interactive and programmatic SQL commands:

```
SELECT NAME FROM VENDOR WHERE VENDOR_NUMBER = '0023'
```


```
SELECT NAME FROM VENDOR INTO :WS-NAME WHERE VENDOR-NUMBER = :WS-VENDOR-NUM
```

In this example, NAME is a column in the table VENDOR. A row with the VENDOR-NUMBER equal to the value of WS-VENDOR-NUM in the COBOL program is selected. And from the selected row, NAME is delivered to the COBOL item WS-NAME. Notice that the only significant difference is the specification of program data names in the programmatic version. (You might also notice the substitution of '-' for ' '. SQL syntax wants an underscore, while COBOL wants dashes, so the preprocessor converts dashes to underscores.)

Additionally, SQL commands have the ability to retrieve multiple records in a single command. This eliminates the need to code loops in many transaction processing programs. (It also speeds up performance, since it reduces the number of entries and exits from SQL.)

Using the sample database, here are code comparisons for IMAGE versus SQL. These statements display a purchase order. The SQL code is actual code, where the IMAGE code is pseudo-code. Notice that the SQL version takes exactly one SQL command to retrieve all qualifying rows.



	HP 3000	<b>DB07/4</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

```
SQL: BULK SELECT * INTO :PO-RECORDS
 FROM PO-HEADER, VENDOR, PO-LINE-ITEMS, PRODUCT
 WHERE PO-HEADER.VENDOR-NUMBER=VENDOR.VENDOR-NUMBER AND
 PO-HEADER.PO-NUMBER=PO-LINE-ITEMS.PO-NUMBER AND
 PO-LINE-ITEMS.PROD-NO=PRODUCT.PROD-NO AND
 PO-HEADER.PO-NUMBER = :WS-PO-NUMBER
```

```
IMAGE: DBGET(MODE7, PO-HEADER, PO-NUMBER, PO-RECORD)
 DBGET(MODE7, VENDOR, VENDOR-NUMBER, VENDOR-RECORD)
 DBFIND(PO-LINE-ITEMS, PO-NUMBER)
 REPEAT
 DBGET(MODE5, PO-LINE-ITEMS, LINE(I))
 DBGET(MODE7, PRODUCT, PROD-NO, PROD(I))
 ADD 1 TO I
 UNTIL (END-OF-CHAIN(PO-LINE-ITEMS))
```

This particular example is a complex one, requiring accesses to four different data sets or tables, and locating multiple records from the PO-LINE-ITEMS dataset or table. Note that the SQL user can test out his/her query interactively, using ISQL, before coding the command.

### Program and Data Independence.

One of the most significant advantages of a database system is the ability to change the database without affecting the executing programs. All database systems have this characteristic to some extent, really none completely implement it. (One example is where a program accesses a field that has been eliminated from a database.)

IMAGE allows addition and deletion of fields of a database by a database administrator. IMAGE allows changing of field definitions by the database administrator such that programs that do not access the changed fields need no modifications.

The mechanism that IMAGE uses to implement this feature is called access by "item list". Specifically, when a program asks IMAGE for data, it presents a buffer, and a symbolic list of data items that describe the items that should fill the buffer. For example, a program might present IMAGE with the item list "VENDOR-NUMBER, VENDOR-NAME".

IMAGE databases are at least initially created by a text file called a "schema". A database administrator creates the schema which defines all sets, items, relationships, and security. Subsequent structural changes can be made to the database by modifying the schema, and recreating the database, or by use of Adager, or a similar utility which recreates the affected datasets. In all cases, the changes are made offline, and the database administrator will probably want to maintain the schema file.

Like IMAGE, SQL provides item flexibility by having programs request data using an item list. SQL also provides a significantly greater degree of program and data independence through a construct call a "View". A VIEW is a logical window that a program uses to access the database. A VIEW might be construed as a logical 'table' in that a program accesses a view just as it might access a table. A view can contain join operations across multiple files.

Here is an example of a VIEW:

VIEW creation:

```
CREATE VIEW PURCHASE_ORDER (PO_NUMBER,VENDOR,AMOUNT) AS
 SELECT PO_HEADER.PO_NUMBER,VENDOR_NAME,AMOUNT
 FROM PO_HEADER,VENDOR
 WHERE PO_HEADER.VENDOR_NUMBER=VENDOR.VENDOR_NUMBER
```

VIEW access (which could be programmatic):

```
SELECT * FROM PURCHASE_ORDER WHERE PO_NUMBER = '1020'
```

The VIEW facility allows external specification of not only the data elements accessed by a program, but also the access path to the data. It allows a program to retrieve data with no knowledge of the access path. It also allows the access path to be changed without requiring alterations to the program.

SQL databases are maintained by SQL commands. These can be given interactively or programmatically, just as any other command. Physical database structure changes can be made while the database is in use. For example, the following command could be given while the specified table is in use:

```
ALTER TABLE VENDOR ADD CLASSIFICATION CHAR(2)
```

This command would add a new column CLASSIFICATION to the table VENDOR. Currently executing programs would not be affected.

The following command could also be given while the database is in use:

```
CREATE INDEX PO_LINE_ITEM ON PO_LINE_ITEMS(PO_NUMBER,PART_NUMBER)
```


This command creates a combined index for the table PO\_LINE\_ITEMS using the columns PO NUMBER and PART NUMBER. Applications using the original database and selecting on PO NUMBER and PART NUMBER would have used the PO NUMBER index, and then searched sequentially for the PART NUMBER. Now those applications can use the new index PO LINE ITEM to go directly to requested line item. This change in access method is transparent to application programs.

### Security.

There is little doubt in the industry today that security is an important job of a database management system. Ad-hoc programs, third-party applications, and open computer systems have mandated externally managed security systems.

IMAGE implements security in the form of passwords. Data items and data sets are passworded for a combination of read/update/none access to data items and read/write/none access to data sets. Passwords are specified by the application program when it opens the database.

Security in SQL is implemented through the granting of access rights to logon user ids. Rather than use a separate password, SQL uses the user logon id, and allows MPE security to be used for passwording. Access is granted against tables or views. Since access to elements can be restricted by using views, data element security is achieved. Hopefully, this will prove to be a simpler technique.

	HP 3000	<b>DB07/6</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

A view, however, is more than simply a subset of data. It can contain not only access specification, but also selection criteria. Since access to data can be granted on views, this allows security to be specified by value. For example:

```
CREATE VIEW P023 AS
 SELECT * FROM PO_HEADER WHERE VENDOR_NUMBER = '0023';
GRANT SELECT ON P023 TO VENDOR23@PURCH;
```

This view allows the user VENDOR23 in the account PURCH to look at only his own purchase orders in the PO\_HEADER table.

### Transaction Management.

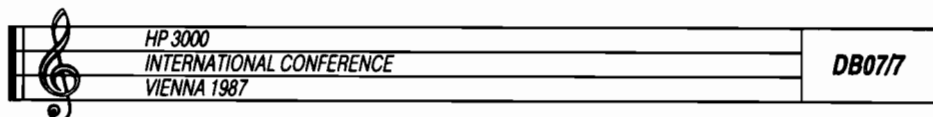
One of the functions of a database management system is to coordinate data between concurrent users. There are two issues: (1) protection against "race conditions" where multiple users desire to access and update the same data, and (2) guaranteeing logical integrity of data. A database management system protects against race conditions by serializing access to the same data. And a database management system guarantees logical integrity by ensuring that either all of its database manipulations succeed, or none of it succeeds.

For example, a user is going to make a transaction which adds one part to inventory, and subtracts one part from a purchase order. The increment to inventory includes reading the data and then updating it. No other updating transaction can be allowed to intervene between the read and update. If an intervening transaction did update the inventory count, then this transaction would make its changes to inventory using the old inventory count, effectively undoing the other transactions inventory update. In the case of system or program failure, the transaction must either have completed, or must be backed out. Otherwise, the partially completed transaction might allow artificial inventory growth.

IMAGE has two facilities to address transaction management: Locking and Transaction Logging. Locking allows programs to logically reserve a specified item before making a transaction against it. Locking is done explicitly by the program. Transaction logging allows a program to declare the beginning and ending of a logical transaction. In the case of system failure, the database can be recovered to last consistent (logically complete) point before the failure. Here is an example of inventory receivings using the sample database:

```
DBLOCK(product.prod-no=2666,po-line-items.prod-no=2666)
DBGET(product,prod-no=2666)
REPEAT
 DBGET(po-line-items,prod-no=2666)
UNTIL (po-number=A2345)
DBBEGIN
 DBUPDATE(product,qty-on-hand)
 DBUPDATE(po-line-items,qty-received)
DBEND
DBUNLOCK
```

In this example, the program locks the item PROD-NO in both the PRODUCT and PO-LINE-ITEMS datasets. Then it retrieves the requisite entries. Once the



entries are found, then a logical transaction is started which updates quantity fields in both datasets.

SQL implements the same constructs, but using a more automated technique. Locks in SQL are implicit; the programmer never needs to code LOCKS into a program. SQL determines concurrency conflicts by examining the data "pages" (which are similar to blocks) accessed within a transaction. If one transaction conflicts with another transaction, then SQL will either wait for the other transaction to complete, or return an error, allowing the transaction to restart itself.

This technique is not only easier to use, but it can also be more efficient. For example, a transaction which updates a bill-of-materials has no idea at the start of the transaction which part-numbers to lock, since the parts-explosion is determined by reading the records to be updated. An explicit locking technique would require either data set locking, or double accesses to the parts dataset. The SQL technique allows maximum concurrency since it does not require pre-determined locking.

Here is the SQL version of the parts receiving problem:

```
BEGIN WORK;
 UPDATE PRODUCT SET qty_on_hand = qty_on_hand + 1
 WHERE prod no = '2666';
 UPDATE PO_LINE_ITEMS SET qty_received = qty_received + 1
 WHERE prod_no = '2666' AND po_number = 'A2345';
COMMIT WORK;
```

SQL assures logical consistency of data using a similar technique to IMAGE: each transaction is bracketted by the commands BEGIN WORK and COMMIT WORK. After a system failure, the uncompleted transactions are backed out in a manner similar to IMAGE. SQL provides the additional feature that if a program aborts, that any incomplete transactions will be rolled back.

A transaction roll-back can also be programmatically initiated by the ROLLBACK WORK command. This feature can simplify, and potentially optimize transaction processing programs. For example: a program to fill sales orders might first match requested line items against inventory to see if the order could be filled. If the order can be satisfied, then it would re-read, and update the records from inventory. The SQL version of this program would simply read and update inventory. If a line item could not be satisfied from inventory, then it would request a rollback.

### In Summary.

SQL provides all of the features of IMAGE, and in most cases in a significantly enhanced fashion. Additionally, SQL allows the user to administer databases at a considerably higher level. With SQL, the database administrator has a high degree of program independent control over data and access paths. In essence, SQL has provided the database administrator with many of the tasks that require programming (and debugging) in IMAGE.

The simplicity of IMAGE has resulted in very good performance for well designed IMAGE databases. IMAGE performance is well understood and reasonably consistent.

On the positive side of performance for SQL is that when a database performance issue arises after an application has already been implemented, that the database administrator can take action without involving changes to program logic. In other words, SQL allows the database designer to make mistakes in the initial design, and correct them after the fact.

#### References.

- Astraham, M. M., et al, "System R: A Relational Approach to Database Management," *ACM Transactions on Database Systems* 1, No. 2 (June 1976).
- Codd, E. F., "Normalized Data Base Structure: A Brief Tutorial", Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control. Available from ACM.
- Codd, E. F., "Further Normalization of the Data Base Relational Model", in *Data Base Systems*, Courant Computer Science Symposia Series, Vol 6, Prentice Hall (1972).
- Russell, Marguerite (ed.), *The IMAGE Handbook*, Wordware(1984), Seattle, WA.
- Also, TurboIMAGE and HPSQL Reference Manuals from Hewlett-Packard Co.

#### Comparison of Limits.


	<u>TurboImage</u>	<u>HPSQL</u>
Sets/Tables per Database	199	Unlimited
Items/Columns per Database	199	Unlimited
Items/Columns per Dataset/Table	255	64
Item Size	4094	3996
Items/Columns per Path/Index	1	15




Sample SQL/COBOL program.

This program prompts the user for a product number, and displays all purchase orders against that product, including which vendor that the purchase order was issued to. Each SQL statement is bracketted by EXEC SQL/END-EXEC, to signal to the pre-processor that these are SQL commands. Notice that the pre-processor also knows the data-division elements, allowing it to check on data types and lengths.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. POS.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 I PIC S9(4) COMP.
EXEC SQL INCLUDE SQLCA END-EXEC. <<SQL communication area>>
EXEC SQL BEGIN DECLARE SECTION END-EXEC. <<SQL data elements>>
01 PURCHASE-ORDERS.
 05 PURCHASE-ORDER OCCURS 20 TIMES.
 10 PO-NUMBER PIC X(6).
 10 VENDOR-NAME PIC X(20).
 10 QUANTITY PIC S9(4) COMP.
01 PROD-NO PIC X(4).
01 ERROR-MSG PIC X(72).
EXEC SQL END DECLARE SECTION END-EXEC.
PROCEDURE DIVISION.
OPEN-DATABASE.
 EXEC SQL WHENEVER SQLERROR GO TO SQL-ERROR END-EXEC.
 EXEC SQL CONNECT TO 'PURCHDB' END-EXEC.
ASK-FOR-PART-NO.
 DISPLAY 'ENTER PROD-NO FOR INQUIRY, OR CR TO STOP'.
 MOVE SPACES TO PROD-NO.
 ACCEPT PROD-NO.
 IF PROD-NO EQUAL SPACES THEN GO TO CLOSE-DATABASE.
 EXEC SQL BULK SELECT PO-LINE-ITEMS.PO-NUMBER,VENDOR-NAME,QUANTITY
 INTO :PURCHASE-ORDER
 FROM PO-LINE-ITEMS,VENDOR
 WHERE PO-LINE-ITEMS.PO-NUMBER=PO-HEADER.PO-NUMBER AND
 PO-HEADER.VENDOR-NUMBER=VENDOR.VENDOR-NUMBER AND
 PO-LINE-ITEMS.PROD-NO = :PROD-NO END-EXEC.
 IF SQLCODE GREATER THAN 0 THEN DISPLAY "NO POS FOR THAT PART-NO"
 ELSE PERFORM DISPLAY-PO VARYING I FROM 1 BY 1 UNTIL I > SQLERRD (3).
 GO TO ASK-FOR-PART-NO.
DISPLAY-PO.
 DISPLAY "PO-NUMBER=" PO-NUMBER (I)
 " VENDOR-NAME=" VENDOR-NAME (I)
 " QUANTITY=" QUANTITY (I).
SQL-ERROR.
 EXEC SQL SQLEXPLAIN :ERROR-MSG END-EXEC.
 DISPLAY ERROR-MSG.
CLOSE-DATABASE.
 EXEC SQL RELEASE END-EXEC.
STOP RUN.
```

	<i>HP 3000</i>	<i><b>DB07/10</b></i>
	<i>INTERNATIONAL CONFERENCE</i>	
	<i>VIENNA 1987</i>	

Larry Kemp is a Systems Consultant for Hewlett-Packard in the Bellevue, Washington (USA) office. Larry Specializes in system performance and data management tools. He has a Master of Science (MS) degree from the University of Oregon (1975). Larry has worked at Hewlett-Packard since March of 1980.

	HP 3000	DB08/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## RELATIONAL DATABASE: HOW DO YOU KNOW YOU NEED ONE?

Orland Larson  
Hewlett-Packard  
Cupertino, California

### ABSTRACT

The field of relational technology is clearly misunderstood by a large number of people. One major obstacle to acceptance of the relational model is the unfamiliar terminology in which relational concepts are expressed. In addition, there are a number of misconceptions that have grown up in the past few years concerning relational systems. The purpose of this paper is to define those terms, correct some of those misconceptions and to help you decide if your company can benefit from adding relational database technology to your current capabilities.

This paper reports on the growing body of knowledge about relational technology. It begins by reviewing the challenges facing the MIS organization and the motivation for relational technology. It then briefly describes the history of relational technology and defines the basic terminology used in the relational approach. This will be followed by an examination of the productivity features of the relational approach and why it should be seen as a complement rather than a replacement for existing network databases such as the IMAGE data base management system. Typical application areas where the relational approach can be very effective will also be surveyed. Finally, a checklist will be reviewed that will help the audience determine if, indeed, they really can benefit from using a relational database.

### INTRODUCTION

#### THE CHALLENGES FACING MIS

The MIS manager is facing many challenges in today's modern information systems organization. The backlog of applications waiting to be developed is one of key challenges concerning MIS. In most medium to large installations the backlog of applications waiting to be developed is anywhere from two to five years. This estimate doesn't include the "invisible backlog," the needed applications which aren't even requested because of the current known backlog. Software costs are increasing because people costs are going up and because of the shortage of skilled EDP specialists. The database administrator typically uses nonrelational databases where a great deal of time is spent predefining data relationships only to find that the users data requirements are changing dynamically. These changes in user requirements cause modifications to the database structure and, in many cases, the associated application programs.



The application programmer is spending a significant amount of time developing applications using these non-relational databases, which require traversing or navigating the data base. This results in excessive application development time. Because the users' requirements change dynamically, a great deal of time is spent maintaining applications. The programmer is also frequently restricted by the data structures in the database, adding to the complexity of accessing data.

End users or business professionals are frustrated by the limited access to information that they know exists somewhere in the database. Their business environment is changing dynamically, and they feel MIS should keep up with these changes. They find that the applications are inflexible, due to the pre-defined relationships designed into the data base. They also lack powerful inquiry facilities to aid in the decision-making process, which would allow them to ask anything about any data residing in that database.

#### THE MOTIVATION FOR RELATIONAL

Dr. Edgar F. Codd, considered to be the originator of the relational model for databases, noted when presented the 1981 ACM Turing Award that the most important motivation for the research work resulting in the relational model was the objective of providing a sharp and clear boundary between the logical and physical aspects of data base management (including data base design, data retrieval, and data manipulation). This is called the data independence objective.


A second objective was to make the model structurally simple, so that all kinds of users and programmers could have a common understanding of the data, and could therefore communicate with one another about the database. This is called the communicability objective.

A third objective was to introduce high-level language concepts to enable users to express operations on large chunks of information at a time. This entailed providing a foundation for set-oriented processing (i.e., the ability to express in a single statement the processing of multiple sets of records at a time). This is called the set-processing objective.

Another primary motivation for development of the relational model has been to make data access more flexible. Because there are no pointers embedded with the data, the relational programmer does not have to be concerned about following pre-defined access paths or navigating the database, which force him to think and code at a needlessly low level of structural detail.

#### THE RELATIONAL DATA MODEL: A BRIEF HISTORY

In 1970, Dr. Codd published an article in the Communications of the ACM entitled "A Relational Model of Data for Large Shared Data Banks." This classic paper marks the "birth" of the relational model. Dr. Codd was the first to inject mathematical principles and rigor into the study of database management.

	HP 3000	<b>DB08/3</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

By the mid 70's, there were two database prototypes being developed. IBM was behind a project called "System R," and there was another relational database being developed at the University of California, Berkeley, called INGRES. It was late 1979 before the first commercially available relational database, called ORACLE, arrived in the marketplace from ORACLE Corporation. ORACLE is also an implementation based on "System R". In 1981 Relational Technology Inc. introduced INGRES which was a different implementation based on the research done at Berkeley. Today there are several additional advanced relational products available, such as HPSQL from Hewlett-Packard, SQL/DS and DB2 from IBM, Rdb from Digital Equipment Corporation and SUPRA from CINCOM. There are additional products sometimes referred to as "born again" relational databases, such as IDMS/R from Cullinet, DATACOM/DB from Applied Data Research and ADABAS from Software AG, to name a few.

#### RELATIONAL DATABASE DEFINED

The relational database model is the easiest to understand - at least at the most basic level. In this model, data are represented as a table, with each horizontal row representing a record and each vertical column representing one of the attributes, or fields, of the record. Users find it natural to organize and manipulate data stored in tables, having long familiarity with tables dating from elementary school.

The Table, or two dimensional array, in a "true" relational database is subject to some special constraints. First, no row can exactly duplicate any other row. (If it did, one of the rows would be unnecessary). Second, there must be an entry in at least one column or combination of columns that is unique for each row; the column heading for this column, or group of columns, is the "key" that identifies the table and serves as a marker for search operations. Third, there must be one and only one entry in each rowcolumn cell.

A fourth requirement, that the rows be in no particular order, is both a strength and a weakness of the relational model. Adding a new record can be thought of as adding a row at the bottom of the table; hence there is no need to squeeze a new row in between preexisting rows as in other database structures. However, to find a particular row, the entire table may have to be searched.

There are three kinds of tables in the relational model: base tables, views, and result tables. A base table is named, defined in detail, filled with data, and is more or less a permanent structure in the database.

A view can be seen as a "window" into one or more tables. It consists of a row and/or column subset of one or more base tables. Data is not stored in a view, so a view is often referred to as a logical or virtual table. Only the definition of a view is stored in the database, and that view definition is then invoked whenever the view is referenced in a command. Views are convenient for limiting the picture a user or program has of the data, thereby simplifying both data security and data access.

A result table contains the data that results from a retrieval request. It has no name and generally has a brief existence. This kind of table is not stored in the database, but can be directed to an output device.

## THE RELATIONAL LANGUAGE

The defacto industry standard language for relational databases is SQL. SQL is pronounced "SEQUEL" and stands for Structured Query Language. This name is deceiving in that it only describes one facet of SQL's capabilities. In addition to the inquiry or data retrieval operations, SQL also includes all the commands needed for data manipulation. The user only needs to learn four commands to handle all data retrieval and manipulation of a relational database. These four commands are: SELECT, UPDATE, DELETE and INSERT.

The relational model uses three primary operations to retrieve records from one or more tables: select, project and join. These operations are based on the mathematical theories that underlie relational technology, and they all use the same command, SELECT. The select operation retrieves a subset of rows, that meet certain criteria, from a table. The project operation retrieves specific columns from a table. The join operation combines data from two or more tables by matching values in one table against values in the other tables. For all rows that contain matching values, a result row is created by combining the columns from the tables, eliminating redundant columns.

The basic form of the SELECT command is:


```
SELECT some data (column names)
FROM some place (table names)
WHERE certain conditions (if any) are to be met.
```

In some instances WHERE may not be necessary. Around this SELECT..FROM..WHERE structure, the user can place other SQL commands in order to express the many powerful operations of the language.

In all uses of SQL, the user does not have to be concerned with how the system should get the data. Rather, the user tells the system what data is needed. This means that the user only needs to know the meaning of the data, not its physical representation, and this feature can relieve the user from many of the complexities of data access.

The data manipulation operations include UPDATE, DELETE and INSERT. The UPDATE command changes data values in all rows that meet the WHERE qualification. The DELETE command deletes all rows that meet the WHERE qualification and the INSERT command adds new rows to a table.

When retrieving data in application programs, it is important to remember that SQL retrieves sets of data rather than individual records and consequently requires different programming techniques. There are two options for presenting selected data to programs. If an array is established in the program, a BULK SELECT can retrieve the entire set of qualifying rows and store them in the array for programmatic processing. Alternatively, it is possible to activate a cursor that will present rows to programs one at a time.

	HP 3000	<b>DB08/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

SQL has a set of built-in, aggregate functions. Some of the functions available are COUNT, SUM, AVERAGE, MINIMUM, and MAXIMUM. They operate on a collection of values and produce a single value.

In addition to commands for data retrieval and modification, SQL also includes commands for defining all database objects. The data definition commands are CREATE, ALTER and DROP. The CREATE command is used to create base tables and views. The ALTER command provides for the expansion of existing tables and the DROP command deletes a view. One of the most powerful features of SQL is its dynamic definition capability. This function allows the user to add columns, tables and views to the database without unloading and reloading existing data or changing any current programs. More importantly, these changes can be made while the databases are in use.

#### PRODUCTIVITY FEATURES OF USING RELATIONAL TECHNOLOGY

Relational technology is one very important tool that can contribute to making data processing professionals more productive. The programmer can benefit from a facility called interactive program development, which allows the development and debugging of SQL commands and then permits the moving of those same commands into the application programs. It is convenient and easy to set up test databases interactively and then to confirm the effect of a program on the database. All of these characteristics make SQL a powerful prototyping tool. The on-line facilities of SQL can be used to create prototype tables loaded with sample or production data. On-line queries can easily be written to demonstrate application usage. End users can see the proposed scheme in operation prior to formal application development. In this prototype approach, people-time and computer-time are saved while design flaws are easily corrected early in development.

The database administrator profits from the productivity features already described for programmers. The database administrator has a great deal of freedom in structuring the database, since it is unnecessary to predict all future access paths at design time. Instead, the DBA can concentrate on specific data requirements of the user. Nonrelational models, on the other hand, require all relationships be pre-defined, which adds to the complexity of the application and lengthens development time.

Additional productivity features for the database administrator include the capability to modify tables without affecting existing programs and the capability to dynamically allocate additional space while the database is still in use. SQL goes far beyond many database management systems in the degree of protection that it provides for data. Views make it possible to narrow access privileges down to a single field. Users can even be limited to summary data. Protection can be specified for database, system catalog, tables, views, columns, rows and fields. It is also possible to restrict access to a subset of commands. These access privileges can be changed dynamically, as the need arises.

In many installations, the key to overall productivity is the ability of data processing to offload the appropriate portions of the development and maintenance to the end user. The flexible design approach of relational databases allows an application to be designed with the end user's requirements in mind. This could enable the DP professional to implement an application up to the point where the end user could create and execute his own queries, thereby expanding the application on his own and reducing his dependence on the data processing department. Through SQL, the end user is provided with extremely flexible access and simple but powerful commands.

**RELATIONAL AND NONRELATIONAL: COMPLEMENTARY TECHNOLOGIES**


Within a data processing department already using a well-established nonrelational DBMS, what role can relational technology be expected to play? We know that DP will not automatically drop everything and go to relational database technology. Rather, relational technology should be seen as a complement rather than a replacement for nonrelational database systems. Both approaches offer a host of benefits, and most applications can be implemented with either of the two.

The relational approach is preferred when the application has a large number of data relationships or when the data relationships are unknown or changing dynamically. The relational approach provides the needed flexibility to establish relationships at the time of inquiry, not when the database is designed. If the application has unknown or incomplete data specifications, which is usually the case in a prototyping environment, then a relational system may be preferable. If the application requires a quick turnaround, the quick design and implementation capabilities of a relational database can be important. The ability to handle ad hoc requests is a definite strength of the relational model as is the ability to extract data for use in a modeling, forecasting, or analytical framework.

The nonrelational approach is preferred for high-volume, on-line transaction processing applications where performance is the most critical requirement.

**CHOOSING THE RIGHT TECHNOLOGY**

The choice of the "correct" database management system must be based on the environment in which the database will be used and on the needs of the particular application. The key feature of relational technology is that it allows for maximum flexibility, and will probably be the choice for many new applications. On the other hand, nonrelational systems may continue to be preferable for very stable or structured applications in which data manipulation requirements are highly predictable, and high transaction throughput is important.

	HP 3000	<b>DB08/7</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The optimum approach for many MIS departments will be to use the relational system concurrently with the existing nonrelational system, matching the appropriate technology to the application. The only problem with such an approach is that the data for an application developed in one technology may sometimes be needed by applications developed in the other technology. Data may be "locked out" from an application that needs it, or users might be tempted to duplicate the data, maintaining both copies. The most desirable solution would obviously be to provide both relational and nonrelational access to a single database. This capability will be available with HP's ALLBASE.

#### RELATIONAL TECHNOLOGY CONSIDERATIONS

There are several things to consider when making the decision to go to a relational database environment. The additional resources usually required to support this technology could significantly impact your system. For example, the intelligence built into the software and the dynamic capabilities of the relational approach usually require additional CPU cycles and memory.


Performance is usually a factor when considering the relational approach and often depends on the maturity of the optimizer software which is built into the relational DBMS. The Data Base Administrator is very important when considering relational and plays a major role in monitoring and improving performance by creating and dropping indexes when necessary. The DBA can also elect to use "clustering" or keeping "like data" together which affects performance by reducing the number of times a disc is accessed.

The command driven nature of SQL may be difficult for some users because they usually have to know the names of the tables and data fields in order to properly construct a SQL command and may prefer a much more "friendly" menu-driven interface. The SQL user must also know the beginning and end of transactions that modify the database and when to "commit work" against that database.

Security of the data resources is usually very important, and the DBA has the capability to implement some very comprehensive security schemes. In addition, to ensure data integrity, logging transactions is mandatory and the user has no way of turning logging off.

If your organization currently has SQL users in an IBM environment they will find little difference in a Hewlett-Packard SQL environment. The user and programmer interface is essentially the same; however, there are some Data Base Administrator functions which are system dependent.

Future releases of HPSQL will work with 4th generation languages and the System Dictionary. In addition, an easy-to-use menu-driven report writer for HPSQL end users and programmers will soon be available.

	HP 3000	<b>DB08/8</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## RELATIONAL APPLICATIONS

There are many application areas - particularly those involving user analysis, reporting, and planning - where the very nature of the application is constantly changing. Some typical application areas are:

- \* Financial
  - Budget analysis
  - Profit and Loss
  - Risk assessment
- \* Inventory
  - Vendor performance
  - Buyer performance
- \* Marketing and sales
  - Tracking and analysis
  - Forecasting
- \* Personnel
  - Compliance
  - Skills and job tracking
- \* Project management
  - Checkpoint/milestone progress
  - Development and test status
- \* EDP auditing
  - Data verification
  - Installation configuration
- \* Government/education/health
  - Crime and traffic analysis
  - Admissions/recruiting/research
  - Medical data analysis

These applications typify instances where it is of primary importance to establish interrelationships within the database and to define new tables.

## CHECKLIST FOR DECIDING WHETHER OR NOT YOU NEED A RELATIONAL DATABASE

Note: If you answer yes to any of the following questions, you should seriously consider taking advantage of relational technology.

1. Does your company have an excessive backlog of applications to be developed, including an invisible backlog?
2. Is your company spending too much money developing applications due to the complexities of using nonrelational systems?

3. Are your programmers spending too much time maintaining applications caused by changing data requirements or relationships?
4. Are your programmers spending an excessive amount of time writing code to navigate through nonrelational databases?
5. Is the nature of your applications constantly changing?
6. Do your users' requirements for information change dynamically?
7. Do your users feel restricted by a nonrelational database?
8. Would your users find it natural to organize and manipulate data in tables?
9. Do your users currently use LOTUS 1-2-3 or spreadsheets?
10. Is your company moving towards a distributed database environment?


#### SUMMARY

Relational technology can have a profound effect on the way organizations operate. In short, the use of relational databases, within the correct DP environment, can help turn the computer into the effective tool most managers need to run their organizations successfully. The following conclusions deal with relational database technology.

- \* Relational concepts are easy to understand and use.
- \* SQL is a multifunctional language.
  - Database definition and creation
  - Data retrieval
  - Data manipulation
  - Authorization and security
  - Transaction management and recovery
  - Database environment management and restructuring
  - Interactive and programmatic use
- \* SQL allows you to specify which information you want - not how to retrieve it.
- \* SQL increases programmer productivity and lifts programming to the level of problem solving.
- \* Data independence is ensured and minimizes maintenance of programs
- \* Data access is automatically optimized as DB structure changes.
- \* The DBA has unprecedented power and control over the database.
- \* New systems are implemented much faster.
- \* Relational databases provide a cost effective powerful solution.


It is to the advantage of most data processing management to learn to use this technology creatively and to manage it effectively. The bottom line is that RELATIONAL DATABASE TECHNOLOGY IS HERE TO STAY!



	HP 3000	DB08/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

#### REFERENCES

- Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," CACM, 13 6,(June 1970),pp. 377-387.
- Codd, E.F., "Relational Database: A Practical Foundation for Productivity," CACM, 25 2,(February 1982),pp. 109-117.
- Date, C.J., An Introduction to Database Systems, Addison-Wesley, 1977.
- Date, C.J., An Introduction to Database Systems Vol II, Addison-Wesley, 1983.
- Schussel, George, "Relational Database-Management Concepts, "Proceedings 1986 Database and Fourth/Fifth Generation Language Symposium, NY, NY, June 8-12, 1986.
- \_\_\_\_\_, Relational Technology: A Productivity Solution, Hewlett-Packard Co., Computer Systems Division, Cupertino, Ca., 5954-6676, January 1986.
- \_\_\_\_\_, SQL/Data System for VSE: A Relational Data System for Application Development, IBM Corp. Data Processing Division, White Plains, N.Y., G320-6590, Feb 1981.

	HP 3000	DB09/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

# Linking To HP System Dictionary

By Ron Harnar  
Hewlett-Packard  
Information Networks Division  
Cupertino, CA 95014

System Dictionary is a new data dictionary product which, for the first time on the HP3000, allows programmatic access to a fully extensible dictionary architecture. Prior to its release on MPE V G.01.04 (T-delta-4), one of System Dictionary's largest "customers" was Hewlett-Packard itself. HP used the System Dictionary intrinsics to develop the definition loaders SDDBD, SDVPD, and SDCONV, a dictionary maintenance and reporting utility SDMAIN, and the definition extractors SDDBC and SDCDE.

These utilities by no means exhaust the possible applications of the System Dictionary intrinsics. But they do embody some dictionary access techniques that will almost certainly be used in every System Dictionary application.

The purpose of this paper is to describe these techniques so that application developers can spend less time reinventing them and more time developing System Dictionary-linked solutions for their target users. This paper starts with a brief overview of the System Dictionary architecture and intrinsics, but a general familiarity with System Dictionary and its utilities is assumed. For details on syntax and usage, refer to the *HP System Dictionary Intrinsics Reference Manual (32254-90002)*.

## Entity-Relationship Model

System Dictionary uses a simple but powerful entity-relationship model in which the central components are entities, relationships between entities, and attributes that define and characterize entities and relationships. You can use the entity-relationship model to describe the real-world objects of an information network, and to define logical connections between these objects.

### ENTITIES AND RELATIONSHIPS

Entities and relationships are the definitions that you store and retrieve in the dictionary. *Entities* are dictionary definitions that refer to objects in the real world of an information network. An entity has attribute values that further define the real-world object or that describe the entity itself (when it was created in the dictionary, who "owns" it, and so on). A *relationship* is an ordered list of entities. Relationships express logical connections between real-world objects. Most relationships involve two entities and are called *binary* relationships. System Dictionary also supports *N-ary* relationships involving up to six entities. Relationships have attribute values that in some cases describe the dictionary definition itself and in other cases describe one of the *entities* in the relationship. Relationship attribute values can thus be used to document an entity in a particular context or usage.

## STRUCTURES

The System Dictionary entity-relationship model provides a set of structures that support the creation and retrieval of entities and relationships. *Entity types* are structures used to define, categorize, and qualify entities. IMAGE-DATABASE, for example, is a System Dictionary entity type used to define entities that describe IMAGE data bases. *Relationship types* are structures for defining, categorizing, and qualifying relationships. IMAGE-DATABASE contains IMAGE-DATASET is a System Dictionary relationship type used to define relationships between IMAGE data bases and data sets.

*Attributes* are structures for defining the attribute values of entities and relationships. IMAGE-DATASET-TYPE, for example, is a System Dictionary attribute associated with the IMAGE-DATASET entity type. CAPACITY is an attribute associated with the IMAGE-DATABASE contains IMAGE-DATASET relationship type. System Dictionary supports six attribute types. Four of the types are fixed-length: *boolean*, a true-or-false value stored in one byte; *character*, an alphanumeric value with a maximum length of 255 bytes; *integer*, a 16-bit or 32-bit binary value; and *floating*, a 32-bit or 64-bit floating-point format. The remaining two attribute types--*alias* and *variable*--are "free-floating" in that they are never assigned to entity types or relationship types, but values for them can be assigned to any entity or relationship. An alias attribute is a 32-byte character field that contains an alias value for an entity or relationship. An attribute of type variable (sometimes called a variable-length attribute) has no maximum length, and is typically used to store descriptions, edit masks, default values, long names and other variable-length values. Since not every entity or relationship needs alias and variable-length attributes, System Dictionary sets aside storage space for them only if they are explicitly assigned. When an alias or variable-length attribute value is deleted, the storage space is released.

A *relationship class* is a name, like *contains* or *uses*, that describes the action or connection of a relationship. In System Dictionary, a relationship type belongs to a relationship class. The relationship types RECORD contains ELEMENT and FORM contains ELEMENT, for example, belong to the contains relationship class. In some cases, a relationship class serves as a qualifier for a relationship type. For example, System Dictionary has three relationship types that involve element pairs:

```

ELEMENT contains ELEMENT
ELEMENT redefines ELEMENT
ELEMENT references ELEMENT

```

The first type is used to create relationships between parent and child entities. The second type indicates that two elements share common storage in a program. The third type documents an element that references another element, as in a Pascal type reference. These ELEMENT ELEMENT relationship types are *qualified* by relationship class.

In System Dictionary, each entity type and relationship type has an attribute list. Attributes are associated with an entity type or relationship type by means of a structural component called a *type-attribute association*. When an attribute is associated with an entity type or relationship type, each entity or relationship of that type must have a value for the attribute.

## Extensibility

System Dictionary provides a built-in set of structures called the *core set*. The core set includes entity types, relationship types, relationship classes, attributes, and type-attribute associations. Users can extend their dictionaries by creating new structures or by making limited changes to the core set such as changing the names of structures. The extensibility feature allows the dictionary to be customized by the user. It also allows Hewlett-Packard to localize System Dictionary to a user's native language, and to update the core set as new subsystems are added to MPE.

## Domains and Versions

In System Dictionary, a single physical dictionary can be divided into multiple, logically separate domains. A domain is a "name space" that contains entities and relationships. Domains are typically used to keep the entity and relationship definitions of one application group separate from others. They can also be used to avoid naming conflicts. Every System Dictionary has a built-in domain called the *common domain* that is always present and always public. Users can create their own domains, called *local* domains, and assign them a sensitivity level of public or private.

Domains can be further subdivided into named partitions called versions. Versions allow you to designate one set of entities and relationships in a domain as the current *production* version, and other sets as *test* or *archival* versions. Unlike domains, which are considered separate name spaces, versions should be thought of as more-or-less complete *copies* of the entities and relationships in a domain. In a version of test status, you can add or delete entities and relationships, or change their attribute values. An archival version cannot be modified, and is usually kept for historical purposes. A production version also cannot be modified, and only one production version is allowed in a given domain. The version feature thus allows you to maintain a stable production version while you experiment with other versions. It also allows you to re-activate an archival version if it becomes necessary to return to a previously used set of definitions.

An important function of a data dictionary is to ensure the standardization and integrity of the definitions used in an information system. The domain and version features allow you to experiment with new definitions and maintain separate sets of definitions. If used carelessly, however, the System Dictionary domain and version features can defeat the standardization value of the dictionary. System Dictionary has security features that prevent unauthorized users from creating domains and versions, and that prevent one user from creating new versions of an entity or relationship owned by another user.

To help overcome the potential integrity problem of having duplicate definitions in several domains, System Dictionary allows an entity or relationship definition in the common domain to be *linked* with one or more definitions in one or more local domains. When definitions are linked they automatically share attribute values except for special attributes such as SCOPE-OWNER and DATE-CREATED. The sharing of attribute values has two advantages. It allows multiple entities or relationships to share the same attribute value storage space, and it ensures that the shared values can only be modified in the common domain, not in the local domains. This gives the owner of the common domain entity or relationship some control over the integrity of the shared attribute values.

## Dictionary Security

System Dictionary provides a comprehensive security scheme consisting of scopes, sensitivity levels, and scope associations.

A scope is a dictionary user name with a password and a set of capabilities called scope rights. The System Dictionary scope rights determine whether a user can create or merely read entities and relationships (the *create* and *read* scope rights), extend the dictionary structure (the *extend* scope right), create scopes and obtain information about dictionary security (the *secure* scope right), and create domains and versions (the *domain* and *version* scope rights).

Scopes "own" entities, relationships, structures, domains, versions, and even other scopes. Some intrinsics require you to be the owner of the dictionary object on which the requested operation (such as deletion) is to be performed. Every dictionary has a built-in scope named CORESET that owns the entity types, relationship types, and other structures of the core set. When you create a dictionary, you specify a name and password for the Dictionary Administrator (DA) scope. The DA scope automatically has all scope rights plus other capabilities not available to other scopes.

Each entity and relationship has a *sensitivity level* that determines whether other scopes can read and/or modify it. Domains also have sensitivity levels that designate them as public or private. If an entity or relationship has a sensitivity level of *read* or *private*, its scope-owner can grant a higher level of access to a selected scope by creating a *scope association*. The scope-owner of a private domain can similarly create a scope association that gives a particular scope access to the domain.

## Intrinsic Groups

System Dictionary has a large number of intrinsics (92). The number is large because the features built into the dictionary architecture--domains, versions, scopes, and an extensible structure--give it tremendous complexity, and because the intrinsics provide comprehensive access to the dictionary. However, since not every System Dictionary application needs every intrinsic, it may help to divide the intrinsics into the task-oriented groups mentioned below.

The *control* intrinsics are used to perform operations such as opening and closing the dictionary and checking for errors. The *entity* and *relationship* intrinsics are used to create, delete, modify, or retrieve entities and relationships. For most applications that access System Dictionary, these will be the most commonly used intrinsics.

The *structure* intrinsics have two general uses. One use is to retrieve information about entity types, relationship types, attributes, and other components of the dictionary structure. This information is typically used to support the creation or retrieval of entities and relationships. A more specialized use (requiring a scope with the extend scope right) is to change the dictionary structure by adding, deleting, or modifying entity types, relationship types, relationship classes, attributes, or type-attribute associations.

The domain and version features of System Dictionary are supported by the *domain* and *version* intrinsics. These intrinsics are used to create, delete, modify, and retrieve

information about domains and versions. They also provide the ability to switch from version or domain to another.

The *security* intrinsics can be used to retrieve information about scopes and scope associations, or to create and maintain a security scheme. The security intrinsics generally require the user to have a special capability (the secure scope right) or owner access to the desired information. This helps prevent unauthorized users from cracking the dictionary security. The scope intrinsics also allow you to switch from one scope to another while the dictionary is open.

## System Dictionary Applications

Although there are many applications for the System Dictionary intrinsics, they generally fall into three categories:

- *Loading* definitions into the dictionary.
- *Extracting* information from the dictionary for use in a subsystem or application.
- *Maintaining* dictionary definitions, structures, domains, versions, and security.

These application types are discussed below.


## Loader Applications

A definition *loader* is a program or routine that uses the dictionary intrinsics to ensure that an entity or relationship is correctly defined in the dictionary. In some cases, this is done by *creating* a new definition with SDCreateEnt or SDCreateRel. In other cases, the loader may *use* an existing compatible definition, or it may *modify* an existing definition if its attribute values are incompatible and the dictionary security allows the modification.

### HANDLING LOADING CONFLICTS

The simplest case of loading a dictionary definition is to create a new entity or relationship where none exists. Often, however, a utility or application program will encounter conflicts because the desired definition already exists, but its attribute values are incompatible with the definition to be loaded. For example, if a program wants to create an entity named ZIP-CODE of entity type ELEMENT with a BYTE-LENGTH attribute value of 9, and the ZIP-CODE entity already exists in the current version with a BYTE-LENGTH value of 5, the program must decide whether to keep the existing definition or replace it with the new one. (Of course, the decision can be postponed by creating the new definition in another version, but someone must eventually decide whether ZIP-CODE has 9 digits or 5.)

One loading problem occurs when an entity or relationship must be modified but the current scope lacks modify access to the definition. The SDModifyEnt intrinsic requires a scope with modify access to the target entity (or SDERR 412 is returned). A scope that lacks modify access to a relationship cannot use SDModifyRel against that relationship (SDERR 612). If the loader "modifies" an entity or relationship by deleting and re-creating it, the security

	HP 3000	DB09/6
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

restriction is even greater: to delete an entity or relationship, you must be its scope-owner (see SDERR 406 and 617). Also, when you delete an entity or relationship, you automatically delete its attribute values and scope associations. When you delete an entity, any relationships involving the entity are automatically deleted also. SDModifyEnt and SDModifyRel leave this network of information intact. For these reasons, you should use "modify" intrinsics instead of the delete-create approach whenever possible.


Another loading problem can arise when the current scope is not the scope-owner of an entity or relationship to be *created*. You may wonder how you can be the owner of something that does not exist yet. The answer is that it may exist in other versions of the current domain. System Dictionary requires that all versions of an entity or relationship in a given domain have the same SCOPE-OWNER attribute value. You therefore cannot create an entity or relationship in one version if it already exists in another version of the same domain and has a different scope-owner. A loader utility should check for this error (SDERR 406 or 617) when creating an entity or relationship.

## HANDLING LINKED DEFINITIONS

When you create or modify an entity in a local domain, you can optionally link it to an entity in the common domain. This is done by passing the common domain entity's name in the *CommonEntity* parameter of SDCreateEnt or SDModifyEnt. From then on, until the local entity is deleted or the link is broken by passing a slash (/) in the *CommonEntity* parameter, the linked entities share all attribute values (except for special attributes such as SCOPE-OWNER and DATE-CREATED), and the values can only be changed by modifying the entity in the common domain. Relationships can also be linked or unlinked by using the *CommonEntityList* parameter in SDCreateRel or SDModifyRel. Because the attribute values of linked definitions can only be modified in the common domain, a loader utility accessing a local domain needs to be aware of this potential conflict (SDERR 421 and 638).

How can you tell if two entities are linked? The answer depends on whether you are in the common domain or a local domain. In a local domain, you can determine the link status by calling SDGetEnt and checking the *CommonEntity* parameter. After a call to SDGetEnt, the *CommonEntity* parameter contains either the name of the linked entity or, if the entity is not linked, blanks. In the common domain, checking for a link is more complicated because a single entity may be linked to one or more local domain entities, and the linked entities may be in any version of any local domain. A common domain entity may therefore have a *list* of linked entities. To retrieve this list, you call the SDGetLocalEntList intrinsic. Similar techniques can be used to determine the link status of a relationship by calling SDGetRel or SDGetLocalRelList.

If you are loading definitions into a local domain, and you want to modify an existing definition that is linked to the common domain, you can use either of two methods. The first method is to remove the link by passing a slash in the *CommonEntity* parameter of SDModifyEnt or the *CommonEntityList* parameter of SDModifyRel. The second method is to keep the link intact, switch to the common domain, and modify the linked definition. To use this approach, you would first call SDGetEnt or SDGetRel and save the *CommonEntity* or *CommonEntityList* parameter so that you know which definition to modify in the common domain. You would then call SDGetVersion to find the name of the common domain version that contains the linked definition. Next, you would switch to this version in the common domain by calling SDSwitchDomain. After switching to the common domain, you would use SDModifyEnt or SDModifyRel to change the desired attributes of the

	HP 3000	DB09/7
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

linked entity or relationship. These changes would automatically be reflected in the linked definition in the local domain. Finally, you would switch back to the local domain and continue the loading process.

Note that there is an important philosophical difference between the two methods of updating linked definitions. When two or more definitions are linked across domains, they are *equivalent* definitions that share a common set of attribute values. When the common domain definition is modified, the local definitions are automatically changed as well. The common domain definition should therefore be modified only if the change is intended to be global for all the definitions linked to it. If the change is intended only for one local domain definition, the common domain link should be broken.

To handle potential conflicts gracefully, then, a loading routine should:

- Check for conflicts after any SDCreateEnt or SDCreateRel call. It may also help to call SDGetEnt or SDGetRel first, to see if the desired entity or relationship already exists before trying to create it.
- Use SDModifyEnt or SDModifyRel to modify an existing definition rather than deleting and re-creating it.
- Be prepared to handle scope conflicts that arise when the current scope lacks modify access or is not the definition's scope-owner.
- Decide whether to handle common domain links by breaking the link or using domain switches.

The loader routine should also have a general error procedure for handling unexpected errors from IMAGE (System Dictionary is built on an IMAGE data base) or the file system.

## Extractor Applications

Now that you have loaded your entity and relationship definitions into the dictionary, along comes an application that looks up the definitions and uses them in a subsystem or application. This type of program is sometimes called an *extractor*, because it extracts information from entity and relationship definitions in the dictionary. Some typical extractor operations include:

- *Generating* data bases, source code, or other subsystem objects according to definitions in the dictionary.
- *Resolving* data definitions at compile or run time by looking them up in the dictionary.
- *Navigating* through a computer system or network guided by dictionary definitions.
- *Defaulting* according to dictionary values rather than hard-coded values.



Dictionary access for an extractor utility is simpler than for a loader. The extractor does not need to be concerned about the scope, domain, and version conflicts that can arise when creating new definitions or modifying existing ones. Read access to already existing definitions is sufficient.

## GENERATING SUBSYSTEM OBJECTS

The main concern of an extractor utility is to provide a correct and consistent translation of System Dictionary definitions into subsystem definitions, objects, or procedures. A COBOL copy library utility, for example, must understand the System Dictionary entities, relationships, and attributes used to define data elements, records, and files. It must also recognize the data definition syntax, reserved words, and restrictions of the COBOL language.

The designer of a System Dictionary-linked extractor utility should examine the core set to see if it supports the definition of objects in the target subsystem. If additional dictionary structures are needed, the utility designer must consider how the structure changes will be distributed to other dictionaries. If Hewlett-Packard agrees to implement a proposed structure change as an enhancement to the core set, it will be distributed to all System Dictionary users worldwide. Otherwise, it becomes the application developer's responsibility to distribute the change.


The designer of an extractor utility is often interested in a specific set of entities that can be identified by the relationships that link them together. An IMAGE data base creation utility, for example, may wish to find all the IMAGE data sets defined in the dictionary that belong to a data base named ORDERS. In System Dictionary terms, this task translates into finding all the relationships of the type IMAGE-DATABASE contains IMAGE-DATASET in which ORDERS is the IMAGE-DATABASE entity. This type of retrieval is done with the SDFindRelList intrinsic. SDFindRelList has an *EntitySearchList* parameter in which you specify a list of entity names to be matched during the search. Unlike SDGetRelList, which returns all the relationships of a specified type, SDFindRelList returns only those relationships that meet the entity list criteria.

## RESOLVING DATA DEFINITIONS

The term *fourth generation language* (4GL) is used to describe a programming language that looks up data definitions in the dictionary while the program is being compiled or run. A 4GL saves the programmer the effort of having to write a detailed working-storage section. The programmer supplies the names of the variables to be used, and the 4GL looks up their data types, lengths, and other information in the dictionary. Also, when a data definition must be changed, the change can be made once in the dictionary and it will automatically be reflected in all the programs that use the definition. In the case of a compiled 4GL, the change is transmitted when the programs are re-compiled. In the case of a run-time 4GL, the changes take effect immediately.

## NETWORK NAVIGATION

A data dictionary can be a central repository of information about a network. Networking applications can rely on the data dictionary to provide current information about network devices, connections, and routing. System Dictionary is designed to provide the functionality of a network dictionary. It can play a central role in the design of networking applications.

	HP 3000	DB09/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## DEFAULT VALUES FROM THE DICTIONARY

A data dictionary can be thought of as a storage facility that allows default values to exist independently of the applications that use them. For example, suppose a company is developing a series of programs to manage inventory information in an IMAGE data base. The data base has an item named AMT-ON-ORDER that specifies the number of units to be ordered for the current month. In each program, the default value for this item is set at 10.

After several weeks of using the new programs, the users decide that they want the default value of AMT-ON-ORDER changed to 5. To implement this change, the programmers must find the programs that assign default values to AMT-ON-ORDER, make the requested change, and recompile the programs.

If, on the other hand, the programs had been designed to check the dictionary for the default value of this item, the change could be made in all the programs by changing a single dictionary definition. Furthermore, the change could be made on-line without having to recompile the programs.

One way to store a default value in System Dictionary is to use the core set attribute DEFAULT. DEFAULT is a variable-length attribute, which means that it can contain a default value of any length or format, and it can be assigned to any entity or relationship in the dictionary.

In the AMT-ON-ORDER example, you might create an entity named AMT-ON-ORDER of type ELEMENT, and assign it a DEFAULT attribute value of 10 (and later change this value to 5). It is then up to the programmers to use SDGetEnt to retrieve the AMT-ON-ORDER entity and check its DEFAULT attribute value.


## Maintenance Applications

Maintenance operations involve the ad hoc creation, deletion, modification, or retrieval of definitions. In a broader sense, maintaining the dictionary also includes creating and maintaining a security scheme, customizing the dictionary structure, and handling domains and versions. It might also involve *merging* the definitions and structures of one dictionary into another.

Unlike a loader utility, which can restrict its operations to a narrowly defined set of entity types, relationship types, and attributes, a maintenance utility usually takes a generic, extensible view of the dictionary. The designer of a dictionary maintenance utility should therefore use the various "list" intrinsics rather than hard-coded lists of definitions or structures. Attributes and their values should also be handled with an extensible routine.

### USING STRUCTURE LISTS

The "list" intrinsics SDGetEntTypeList, SDGetRelTypeList, SDGetRelClassList, and SDGetAttrList allow the programmer to retrieve lists of dictionary structures. The lists can be presented to the end user for selection of a structure. Also, since System Dictionary allows the dictionary structure to be extended, programmers can use the list intrinsics to

	HP 3000	DB09/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

keep up with these changes. The maintenance and reporting utility SDMAIN makes extensive use of the list intrinsics.

One example of list intrinsic usage is the task of finding all the relationship types in the dictionary. Since every relationship type belongs to a relationship class, the first step is to call SDGetRelClassList to get the list of relationship classes available in the dictionary. Then, for each relationship class, SDGetRelTypeList would be called to retrieve all the relationship types of that class. SDMAIN uses this two-stage retrieval method when you give the command:

```
>DISPLAY RELATIONSHIP-TYPE.
```

Because you omitted the RELATIONSHIP-CLASS parameter in this command, SDMAIN must call SDGetRelClassList to get the list of classes before it can get any relationship types. If you specify a relationship class, as shown here,

```
>DISPLAY RELATIONSHIP-TYPE; RELATIONSHIP-CLASS=CONTAINS.
```

SDMAIN would skip the SDGetRelClassList intrinsic and go directly to SDGetRelTypeList for the list of relationship types belonging to the contains relationship class.

Another example of the list intrinsics is found in the attribute list prompting feature of SDMAIN. When you create, modify, or retrieve an entity in SDMAIN, you are given a list of attributes to choose from. The attribute list is assembled by calling SDGetEntTypeAttrList to get the list of attributes associated with the specified entity type, and SDGetAttrList to get the list of alias and variable-length attributes available in the dictionary. The complete attribute list presented to the user thus includes not only the entity type attributes but also the "free-floating" alias and variable-length attributes that can be assigned to any entity.

## EXTENSIBLE ATTRIBUTE ROUTINES

A loader or extractor utility is usually interested in a fixed list of attributes, so it is a simple matter to devise a data structure for their attribute values. For maintenance utilities in which the user is allowed to *specify* an attribute list, however, a more generic and extensible method of handling attribute values is needed. The following discussion focuses on the retrieval of *entity* attribute values, but the same techniques can be used for relationships. Consider this example of entity retrieval in SDMAIN:

```
>REPORT ENTITY LAST-NAME; ENTITY-TYPE=ELEMENT;
>>LIST=SCOPE-OWNER, ELEMENT-TYPE, BYTE-LENGTH.
```

With this command, the user is requesting the SCOPE-OWNER, ELEMENT-TYPE, and BYTE-LENGTH attribute values of the LAST-NAME entity. One way to handle this request would be to retrieve each attribute value separately. First you would parse the list parameter to get the attribute names. Then, for each attribute, you would call SDGetAttr to determine its data type and length. Finally, you would call SDGetEnt supplying the attribute name in the *AttributeList* parameter and a variable of the appropriate type and length in the *AttributeValues* parameter. From a performance perspective the use of multiple SDGetEnt calls is expensive, but it makes for a simple looping algorithm that can

handle any number of attributes with a fairly small *AttributeValues* parameter (the maximum length of any single fixed-length attribute value is 255 bytes).

Another method is to retrieve all the requested attribute values in a single SDGetEnt call. In this example, you would simply build an *AttributeList* parameter like the user's list parameter. The difficulty of this method is in building the *AttributeValues* parameter. Since the user can specify any number of attributes, and the attributes can have a variety of data types and lengths, you would need to build a table of attribute information containing each attribute's data type, length, and position in the *AttributeValues* array. Retrieving a list of attribute values in a single SDGetEnt call requires more programming effort, but it is more efficient than the looping method described earlier.

You can assign arbitrary maximum lengths to the *AttributeValues* array and to your attribute information table, but a truly extensible entity retrieval routine should rely on System Dictionary maximums. The *AttributeValues* array should have enough room for all special attributes, all alias attributes (since these can be assigned to any entity), and all other attributes that could be assigned to the entity type via a type-attribute association. The special attributes require 102 bytes. The maximum number of alias attributes allowed in the dictionary (128) times their length (32 bytes) would add 4096 bytes to the array. The maximum length of other fixed-length, non-alias attribute values is 2048 bytes. The maximum length of the *AttributeValues* array should therefore be:

$$102 + (128 * 32) + 2048 = 6246 \text{ bytes}$$

The length of the *AttributeList* parameter should also reflect the dictionary maximums. There are six special attributes assigned to each entity type. Each entity type can have up to 128 additional type-attribute associations. As already mentioned, there can be up to 128 alias attributes in the dictionary, and any number of these can appear in the attribute list. The *AttributeList* array should therefore have room for:

$$6 + 128 + 128 = 262 \text{ attributes}$$

The *AttributeList* parameter can be defined as an array of 32-byte attribute names. It can also be defined as an array of 32-bit integers representing the *internal numbers* of the attributes. Internal numbers are retrieved from the attribute intrinsics (e.g., SGetAttr and SDGetAttrList). When you pass an array of internal numbers in the *AttributeList* parameter, the first 32-bit integer is used as a count item that indicates the number of attributes in the list. This format would therefore require a maximum of 263 array elements. Besides saving space in the *AttributeList* parameter, the use of internal numbers also improves performance by saving System Dictionary the overhead of attribute name lookups.

System Dictionary provides a flexible entity-relationship model and a set of environmental features--domains, versions, and scopes--that allow the dictionary to be tailored to application requirements. The System Dictionary intrinsics provide comprehensive access to entities, relationships, domains, versions, scopes, and the dictionary structure. You can use the intrinsics to design a wide variety of applications that load, extract, or maintain dictionary definitions. This paper discussed some guidelines and techniques for implementing a System Dictionary-linked application.

## Table of Contents

<b>Entity-Relationship Model</b> . . . . .	1
<b>Entities and Relationships</b> . . . . .	1
<b>Structures</b> . . . . .	2
<b>Extensibility</b> . . . . .	3
<b>Domains and Versions</b> . . . . .	3
<b>Dictionary Security</b> . . . . .	4
<b>Intrinsic Groups</b> . . . . .	4
<b>System Dictionary Applications</b> . . . . .	5
<b>Loader Applications</b> . . . . .	5
<b>Handling Loading Conflicts</b> . . . . .	5
<b>Handling Linked Definitions</b> . . . . .	6
<b>Extractor Applications</b> . . . . .	7
<b>Generating Subsystem Objects</b> . . . . .	8
<b>Resolving Data Definitions</b> . . . . .	8
<b>Network Navigation</b> . . . . .	8
<b>Default Values From The Dictionary</b> . . . . .	9
<b>Maintenance Applications</b> . . . . .	9
<b>Using Structure Lists</b> . . . . .	9
<b>Extensible Attribute Routines</b> . . . . .	10

## Is There Life Besides IMAGE?

May Kovalick

Hewlett-Packard  
Information Technology Group  
Cupertino, California, USA

### Introduction

"Knowledge is of two kinds: we know a subject ourselves, or we know where we can find information upon it." - Samuel Johnson, 1775.

Samuel Johnson is the originator of the first English language database. We all have a copy of a similar database on our desk today - the dictionary. In 1775, to have that database at one's disposal was rare and privileged. Today, the possession of such is commonplace. The body of knowledge that existed then was minuscule compared to that of today. Nonetheless, the amount of knowledge or information that we know ourselves is becoming smaller compared to that we have access to. Fortunately, technology is on our side for managing the ever increasing amount of data.

For most of the HP3000 users, the IMAGE database management system has been the tool for storage and retrieval of data for many years. With the advent of more sophistication in the usage of databases, and the increased emphasis on productivity and flexibility, the offering of the relational technology on Hewlett-Packard's family of computers is a must.

The new ALLBASE product is Hewlett-Packard's advanced database management system for the Hewlett-Packard Precision Architecture systems for both the commercial and technical markets. It combines both relational and network model data access in a single product (See Figure 1). HPIMAGE is the IMAGE like interface, and HPSQL is the relational interface that uses the de facto industry standard SQL (Structured Query Language) for both data definition and data manipulation. The co-existence of both interfaces in one database management system allows the user the flexibility to choose the appropriate data model for each database application.

The rest of this paper will concentrate on the HPSQL interface. I will give an overview of the basic data definition and data manipulation functions of HPSQL, describe the major components of HPSQL, highlight some of the features that are unique to Hewlett-Packard's SQL, and give a preview of future directions of Hewlett-Packard's database product offerings.

### HPSQL

HPSQL is a family of relational products available on the different Hewlett-Packard computers:

- HPSQL/V is available on the Series 70 and all previous MPE-V based HP3000 systems.
- HPSQL/XL (a component of ALLBASE/XL) will be available on the HP3000 900 series systems.

- HPSQL/HP-UX (a component of ALLBASE/HP-UX) will be available on the HP9000 800 series systems.

HPSQL provides all the advantages of relational technology. It is easy to learn and use, and provides a set of powerful commands for data definition, data manipulation, security and authorization control, transaction management and database administration.

The implementations of all the HPSQL products are highly leveraged and are totally compatible with one another. Customers may develop applications on one system and be able to move those applications to another without source code modifications.

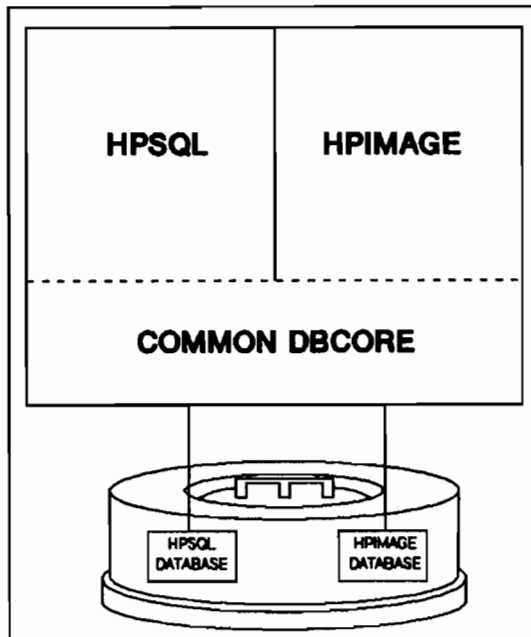



Figure 1. ALLBASE Product Structure

## Data Definition

An HPSQL database is a collection of database objects consisting of tables, views and indexes. A *table* consists of columns and rows, and may be created with the CREATE TABLE command. The relationships among tables are determined, *not* by explicit pointers, but by the data values in the columns of the tables themselves.

A *view* is a virtual table derived by a data manipulation statement from one or more physical tables or views. Views provide some data independence from certain changes to the database. If the user wishes to condense multiple tables into one, or split one table into many, views can protect the user from modifying programs that read these tables.

	HP 3000	<b>DB10/3</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Another use of views is for security. If the user wishes to restrict access to data based on content, a view may be defined to perform value based security checking.

A user may create *indexes* on a table to reduce the time it takes to retrieve data from it. Unlike HPIMAGE application programs where an access path has to be explicitly specified for each database retrieval operation, HPSQL application programs do not specify indexes to be used for the query commands. HPSQL automatically analyzes data access requests in terms of the indexes available and chooses to use the one that will optimize performance.

Tables, views and indexes may be added or deleted dynamically to the database while it is in use. An existing table may also be expanded by the addition of one or more new columns. These dynamic data definition capabilities of HPSQL allow the users to restructure the database easily to reflect changing needs.

## Data Manipulation

HPSQL provides the SELECT, INSERT, UPDATE, and DELETE commands for data access and modification.

The SELECT command in HPSQL allows users to perform the three basic relational retrieval operations of selection, projection and join. Selections produce a horizontal subset (of rows) in a table that satisfies certain criteria. Projection produces a vertical subset (of columns) in a table. Join combines data from two or more tables by matching values in a column of one table with values in a comparable column in the other tables. In addition, the SELECT command supports arithmetic expressions, sorting, grouping operations and a set of built-in aggregate function such as MIN, MAX, AVG, etc.

The INSERT command allows users to add one or more rows to a table. The UPDATE command allows users to modify values in one or more rows of a table. The DELETE command allows users to delete one or more rows from a table.

Using these four basic data manipulation commands, the users can easily specify what data to access or modify without having to specify how to do it.

## Components of HPSQL

Figure 2 shows the architecture of HPSQL. There are five major components:

- The interactive user interface (ISQL).
- The utility package for performing database administration tasks (SQLUtil).
- The preprocessors that provide programmatic access to the database (Preprocessors).
- The parser and query processor (SQLCore).
- The kernel database access module (DBCORE).

I will describe these components from the bottom up.



## DBCore

DBCore comprises the command executor and the low-level services. The command executor is a single, cleanly defined interface point that accepts commands from the interfaces above and calls the low-level services to perform the tasks. The low-level services include all the routines to store, access and update data, and provides transaction management, multi-user concurrency control, logging and recovery.

DBCore does not presume the relational, nor any, model of data. It handles data in a model independent manner. It is this feature that allows both the HPSQL and HPIMAGE interfaces to be built on top of it.

This layer of the software is most dependent on the operating system, especially the modules for accessing files. However, these OS-dependent routines are well encapsulated and insulated, thus making the operating system transparent to the HPSQL and HPIMAGE interfaces above. This modularity allows ALLBASE to be easily portable to the different operating systems.

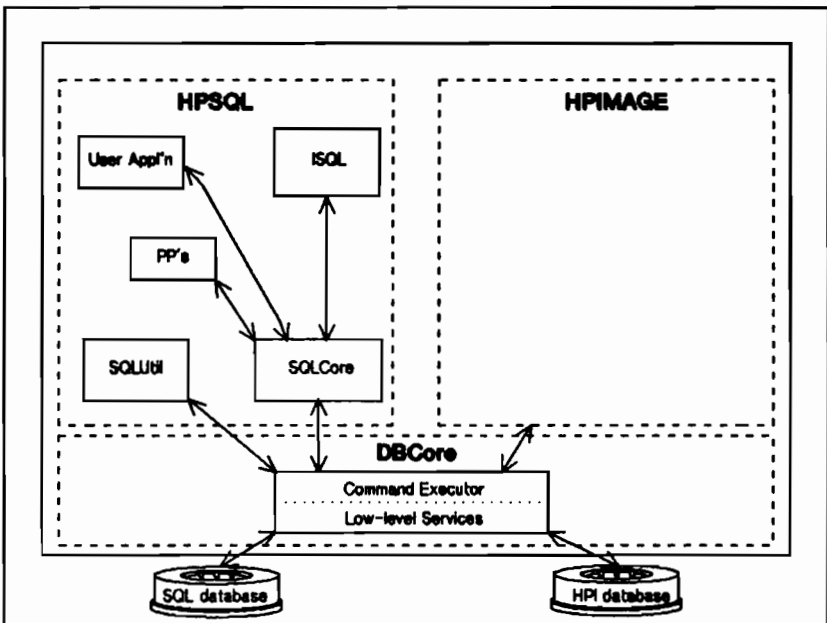



Figure 2. HPSQL Components

27

## SQLCore

SQLCore comprises the parser and the query processor. The parser parses SQL commands and generates command trees which are "flattened", i.e. all the internal machine-dependent pointers are replaced by a machine-independent linear representation. The linearized command trees are then passed on to the query processor.

	HP 3000	<b>DB10/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The query processor performs protection validation, query optimization and access path selection. It also makes sure that the tables and columns referenced in the query are valid, and generates the appropriate DBCore commands to execute the query.

The set of powerful data manipulation commands allows the user to specify what data is to be accessed or modified, but not necessarily how to access them. A query optimizer is contained in the query processor to look at the query and evaluate the current physical structure of the database to determine the most optimal path to access the data. This, however, does not mean that the user has no control over the performance of his queries. Because of the powerful data definition capabilities provided by HPSQL, the user can tune the performance of his/her applications by creating and dropping appropriate indexes for the tables, or by changing the physical configuration of the database.

One such example is the use of a clustering index for a table. When a row is inserted or updated into a table that is defined with a clustered index, HPSQL will attempt to place that row on the same or consecutive data page with other rows with similar key values. Because the rows are physically close, I/O overhead is reduced and performance may be improved whenever the rows are retrieved in key order.

## Preprocessors


HPSQL provides users programmatic access to HPSQL databases via preprocessors. The preprocessors are programs that read the source code of user application programs which have SQL commands embedded in them. The preprocessor looks for the predefined directives (EXEC SQL) in the source programs that define access to HPSQL and replaces them with language specific calls to HPSQL. It also performs optimization for the queries and stores the predefined database commands in the database so that, when the program is executed, the preprocessed commands are executed.

There are two ways that database management systems can allow application access to a database: preprocessors and intrinsics (e.g. DBGET or DBPUT for IMAGE). There are a number of advantages in using the preprocessor approach. Firstly, a preprocessor is usually more friendly than intrinsics. It can take care of data type conversions, language-dependent and OS-dependent calling conventions, and error handling in a manner that is transparent to the user. Secondly, the preprocessor approach improves performance by allowing query optimization to be performed when the application is preprocessed instead of at run-time. At run-time, HPSQL will detect if a change in the database structure has invalidated the access strategy for any of the queries and will automatically re-process those queries for the new structure. Thirdly, there is the advantage of being compatible with other industry implementations and thus provides portability for SQL applications.

The Pascal and COBOL preprocessors are available with HPSQL/V and HPSQL/XL. The Pascal, FORTRAN and C preprocessors are available with HPSQL/HP-UX.

## SQLUtil

SQLUtil provides a set of commands for the database administrator to perform various administrative tasks, such as altering the configuration of the database environment, managing database files, and backing up and restoring the database environments, etc. It can be invoked either from ISQL or from the operating system.

	HP 3000	DB10/6
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## ISQL

ISQL is the interactive user interface that provides the user with functionally complete interactive SQL access to the data. It accepts user SQL commands, sends them to the parser, then passes the "flattened" command trees to the query processor for processing.

ISQL accepts commands from three sources: the terminal, the command buffer, or a command file. The *command buffer* is an area for holding one or more commands for the duration of an ISQL session. The contents of the command buffer may be changed, kept in a file or executed. A *command file* is a system file that contains one or more SQL commands. It may be created outside the ISQL environment, using an editor or a program written by the user.

ISQL also provides a *command history buffer* for holding the 10 most recently submitted commands. Any of the commands from the command history buffer may be listed, recalled for re-execution, or edited and re-executed.

ISQL is a useful facility for different types of users. Frequent users may use it for ad hoc retrieval and modification of data with the usual data manipulation commands. It can be a program development tool for application programmers to build test databases, and to try out queries to be embedded in applications. It is also a tool for database administrators to create and maintain databases, to load/unload data from/into external files, and to define and control physical storage for the databases.

## HPSQL Unique Features

Most of you may already be familiar with HPSQL/V or the industry standard implementation of SQL and the basic functions provided by SQL. For more information, you may refer to the SQL Reference Manual.

In this section, I would like to highlight a few of the SQL features that are unique to the HPSQL product. These features include the following:

- Authorization Groups for security management
- Savepoints for transaction management
- Logging and Recovery
- Bulk Table Processing through the Programmatic Interface

## Security Management and Authorization Groups

A major function of a database administrator is security management, i.e. controlling access to a database and its objects such as tables, views, etc. Since every user must have appropriate authorization in order to access the database and perform operations, the DBA can use HPSQL authorization to maintain security for a multi-user database environment.

An authority is a privilege given to a user or a group of users to access the database environment, create database objects, perform a specific operation, preprocess and run



application programs containing SQL commands, or maintain the database environment. Authorization is provided for the tables, views, and resources of the database. The owner (initially the creator) of an object can perform all operations on the object and can grant authorization to another user to operate on the object. Authorities on an object may be revoked, and ownership of an object may be transferred to another user or group of users.

For most business organizations, a database is shared by multiple departments having different types of access and operations performed on the database. For example, a personnel database may be accessed by different departments such as accounting, payroll, personnel clerks, personnel managers, etc. Each of these departments may have one or more users, and each of the departments may require different access rights to the database. In order to make it easier for the DBA to establish authorization for the database for groups of users according to their database requirements, HPSQL provides a unique feature called *authorization group*.

An authorization group is a group of users that possesses the same set of authorities. A user can be a member of any number of groups, and groups can also be members of other groups. Authorization groups may be created or dropped dynamically. Once a group is created, individual users or groups can be added to an authorization group or removed from the group. When a user is added, he or she automatically acquires the authorities belonging to the group.

The GRANT and REVOKE commands allow the user to specify a group as the receiver of the authority. These authorities then belong to the group and not to the individual members of the group. That is, as long as a user is a member of a group, the user possesses the authorities belonging to the group. If the user is removed, he or she no longer possesses the authorities of the group.

Authorization group is therefore a valuable security management feature for the database administrator to easily control database access for groups of users.




## Transaction Management and Savepoints

A transaction is a unit of work specified by a sequence of SQL commands. A transaction is started by the command BEGIN WORK and ended with the command COMMIT WORK, in which case all changes made by the transaction become permanent. The transaction may be aborted with the command ROLLBACK WORK, in which case none of the changes are made to the database. Most of the commercial SQL implementations provide the above facilities for managing user transactions.

In addition to the above, HPSQL supports savepoints within transactions to allow users to rollback some of the changes in a transaction. A *savepoint* defines a set of commands within a transaction that can be aborted without aborting the entire transaction. A savepoint is defined using the SAVEPOINT command. The user can then undo the changes within a transaction since the savepoint was defined by using the ROLLBACK WORK command. Multiple savepoints can be defined within a transaction and are referred to by a number returned by the query processor in the SAVEPOINT command.

A savepoint may be used in a long transaction that does several operations, some of which might have to be rolled back. Savepoints can greatly reduce the number of transactions that have to be resubmitted because part of the transaction was unsuccessful.

	HP 3000	<b>DB10/8</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Logging and Recovery

In order to support concurrency and still provide data integrity and reliability, HPSQL provides extensive logging and recovery features.

The most common form of damage to a database occurs when a user or system process fails during the execution of a transaction and is unable to complete it, thus rendering the data inconsistent. This is called a soft failure since the database is not seriously corrupted and can potentially be repaired without requiring complete restoration of the data.

HPSQL logs all changes to the data of a database in its log file. Should a soft failure occur, HPSQL will automatically attempt to bring all data back to a consistent state with the information recorded in the log file. All transactions which successfully committed prior to the crash will be recovered. Transactions which failed to complete prior to the crash will be rolled back, or undone. This is called *rollback* recovery. Rollback recovery is automatic and is always available.

A second form of failure results from a hard failure which renders the database unreadable or completely corrupted. Such failures may be due to hardware problems, such as a disc head crash or an operating system error that allows random data to be written on a table in the database.


Should a hard failure occur, it is necessary to restore a stored copy of the damaged database(s) and then roll-forward, or redo, all transactions that were committed before the hard crash and since the stored copy was created. This is called *roll-forward* recovery.

To support roll-forward recovery in HPSQL, an *archive* mode of logging is provided. When HPSQL is run in archive mode, all changes to the database are logged and the log space is never reused. To perform roll-forward recovery, an old copy of the database is restored from a backup or archive copy. The current log contains all the changes since the last backup. Using the START DBE ... RECOVER command, the database is recovered to a consistent state by incorporating all work done by transactions committed before the failure, and excluding any changes made by transactions that did not commit by the time of the failure. A date and time may also be specified in this command for recovering the database to the desired date and time.

A *dual logging* option is available in HPSQL to further enhance integrity. Two separate logs on separate media are maintained. Both logs are written for all operations. Normally only one log is read during recovery, but if an error is encountered, HPSQL switches to the other log automatically. Data integrity is maintained, provided that there is at least one good copy of each log record on either of the logs.

## Programmatic Interface and Bulk Table Processing

HPSQL provides the full set of data definition, data manipulation and transaction management commands through the programmatic interface. This includes the use of host variables, indicator variables for handling null values, run-time error checking and handling, the use of cursors, and dynamic query processing for executing SQL commands that cannot be defined until run-time.

	HP 3000	DB10/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The data manipulation commands in HPSQL allow the user to insert, delete, update and select rows from a database. A single row or multiple rows can be operated upon with one data manipulation command. A cursor may be used to operate on a multiple-row query result, one row at a time. Like the cursor on a terminal screen, an HPSQL cursor is a position indicator. It allows the user to move through the multiple-row query result, retrieving a row at a time into host variables and optionally updating or deleting the row. Reporting applications may find this technique useful.

In addition to the above, HPSQL provides a unique feature in the programmatic interface for *bulk table processing*. The user may specify an application program to retrieve or insert multiple rows with the execution of a *single* SQL command. Three bulk commands are available:

- The BULK SELECT command can be used when you know in advance the maximum number of rows in a multiple-row query result, or when the query result is not too large. For example, an application that retrieves a query result containing a row for each month of the year might find this command useful.
- The BULK FETCH command can be used to handle large query results or multiple-row query results whose maximum size is unpredictable. If a single execution of the BULK FETCH command does not retrieve the entire set of query result, it may be re-executed to retrieve subsequent rows in the query result. This use of a cursor is most suitable for display-only applications, such as programs that allow a user to browse through a query result, so many rows at a time.
- The BULK INSERT command can be used to insert multiple rows into a table. Rows are inserted from a host variable declared as an array.

In the bulk retrieval commands, the user may specify the maximum number of rows to be retrieved and where to put the data. Rows are retrieved into a host variable declared as an array. HPSQL fetches as many rows as will fit in the retrieval area (or the specified maximum number of rows, or the number of rows remaining in the query result, whichever is less). A value is returned telling the user the actual number of rows fetched.

The BULK SELECT command minimizes the time a table is locked for the retrieval operation, because the program can execute the BULK SELECT command, then immediately terminate the transaction, even before displaying any rows. Similarly, the BULK INSERT command is efficient for concurrency, because any exclusive lock acquired to insert rows need be held only until the BULK INSERT command is executed.

The set of bulk table processing commands provided by HPSQL is very valuable for application builders. It provides a set of features complementary to the row-at-a-time cursor operation commands. It allows application programmers easy access and handling of multiple-row data and query results. It also provides good performance for applications that need to handle large amounts of data efficiently.

## Future Directions

Many of Hewlett-Packard's customers have invested heavily in developing database applications using IMAGE. With the introduction of Hewlett-Packard's Precision Architecture systems, many of these customers may choose to migrate their existing

database applications to ALLBASE. HPIMAGE certainly provides an easy and logical migration path. However, users may want to take advantage of the relational technology to increase their productivity and to ease their programming effort.

In view of this, Hewlett-Packard plans to provide users the capability of accessing HPIMAGE data using HPSQL in future releases of ALLBASE (Figure 3).

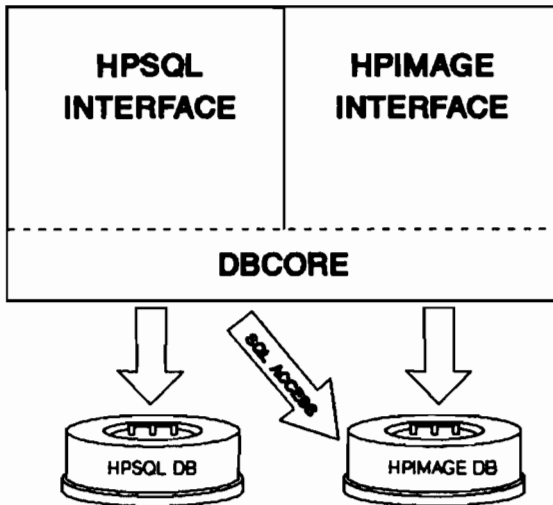


Figure 3. SQL Access to HPIMAGE Data

..


Users may be able to develop new applications against existing HPIMAGE databases using the HPSQL programmatic interfaces. The same data may also be accessed on an ad hoc basis using the powerful and flexible interactive query capability provided by ISQL.

This integration will thus allow users to gain the optimal benefits of both technologies without introducing data redundancy or inefficiencies into the MIS environment.

Another Hewlett-Packard long term goal is to provide its customers with the hardware and software needed to support distributed database management systems. One important step toward that goal is to provide the same powerful database management software on all computers so that data can easily be shared. By offering its customers SQL, the de facto industry standard for relational technology, Hewlett-Packard is moving toward compatibility not only between its own computer systems, but also with those of other suppliers.

## Conclusions

Yes, with ALLBASE, there is life besides IMAGE. The HPSQL and HPIMAGE interfaces serve as complements for each other in ALLBASE. The architecture of ALLBASE is modular, allowing for changing architecture (at the bottom) and for additional user interfaces (on the top). The system architecture of ALLBASE provides a solid foundation to carry Hewlett-Packard's database management plans well into the next decade.

	HP 3000	DB11/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## PREFACE

### HPSQL IN PRACTICE

Andre Van Aken, Sr.Systems Engineer, HP Belgium

Relational Data Bases are in. Many theoretical discussions have already been devoted to this subject. But although HP announced his contribution to the relational data base world, HPSQL, a year ago, there is still very little experience with this product in the HP 3000 world, compared to the vast knowledge of IMAGE/3000. But many HP-customers are interested in HPSQL, for a variety of reasons:

- 1- for some applications, HPSQL offers the flexibility in structure and access-types that is lacking in a network type DBMS like Image.
- 2- customers want to be prepared for ALLBASE, the combined Image/SQL DBMS for HP-Precision Architecture machines.
- 3- Many big companies are moving to DB2 on their IBM-mainframes, and want the same functionality on their departmental systems.
- 4- SQL is an excellent 4th generation tool, i.e. it is a real non-procedural language.

This paper attempts to give some practical advise for the design of applications founded on HPSQL. We will cover 3 out of many possible topics. We will discuss consecutively the logical data base design, the steps involved in converting from Image to HPSQL, and finally the most important differences in program design between those two DBMS-systems.



Much more than in Image, it will be important to fully normalize data structures before you start the physical data base design. The correctness of query-results will depend on the logical correctness of the table-layouts (for instance, are all columns functionally dependant on the key?).

The normalization process can be a very cumbersome task, especially for big data bases with many elements. Therefore there is certainly a need for formal methods for the design of logic data base structures. Maybe there are many alternatives available, but I will mention one method that I know and used myself some years ago. The method calls Nyssen Information Analysis Method(NIAM), which is a formal way of describing entities and relationships. NIAM has rules for creating fully normalized data structures from those binary relations.

Two computer-assisted products exist that accept the NIAM binary sentences as graphic input (so it is a CAD/CAM for data bases), perform automatically the whole normalization process, and are even capable of generating SQL - syntax for immediate data base creation.

One product is QINT/TINA, from QINT Systems, which runs on IBM-PC/AT and compatibles (i.e.the Vectra). The other product is under development at the university of Hasselt (Belgium) and runs on the HP9000/300 series, and uses all of the graphic capabilities of the HP9000 workstation (ref. dept. of professor Meersman).

Probably there are other packages that support the data base design process. The important message is that the era of empiric methods comes to an end once you are going to implement relational data bases.

Concerning the conversion from Image to HPSQL many things are already said. The *HPSQL Data Base Administrator Guide* devotes a whole section on the subject, and in *Interact Magazine* of august '86, Michele Dingserson described a very usefull *step-by-step* process for translating an Image schema to an HPSQL-structure.

I plan to add some extra hints to that, and I will illustrate some rules of thumb for moving data from an Image Data Base to HPSQL with little or no programming effort.

When we have to transform an Image-schema to HPSQL, it is wise to keep in mind the rules and methods of normalization. Some reflection on the logical correctness of the existing Image structure will be indispensable: at least we need to have our data base in first normal form, i.e. we have to split of all repeating groups. For instance a "monthly- sales-turnover" array-item in an orders-data-set, should go to a separate table with perhaps product-no , month, and turnover of the month as columns. This step is necessary since HPSQL does not support array- items by definition.

To decide whether or not to perform further normalization we'll have to compare the benefits of a sound data base structure with the advantage of a close resemblance to the Image schema.

Concerning this Image-to-SQL schema mapping, other sources can be consulted (cfr.the Interact article); therefore I'll concentrate on methods for loading data from one data base to the other, preferably without special programming effort.

Well, with QUERY, or even better with the new Business Report Writer, together with the sophisticated load/unload functions of the ISQL- utility, we have some excellent tools for this job.

ISQL, the utility for interactive acces to HPSQL, allows, via the LOAD-command, to upload an SQL-table directly with data from an external (MPE) file. There is one limitation, though: this file must only contain ASCII values, since ISQL will try to do the proper conversions for integer and decimal column-types.

So, if we have an Image data set, without binary or packed fields, which can map directly to an SQL-table, the QUERY-functions "FIND" (to retrieve all records) and SAVE can provide us with such an external file.

If SAVE would not help, for instance if the set contains integers, then a QUERY-report procedure without headings and edit masks, and with formal file designator QSLIST pointing to a discfile, will do the job. The lucky guys that have BRW on their system, can produce reports with all the desired information, and define BRW-output-files for all SQL- tables that have to be loaded.

Sometimes two Image sets have to be combined into one SQL-table, or one set has to be split in several tables, for the sake of normalization or performance. In this case we can build some temporary SQL-tables which match the Image-sets. After loading the data from the Image data base into those worktables, we can use the powerful unload function of ISQL [cfr. figure 1] to combine the temporary tables to a file that will contain the values for all columns of the final table. This can be achieved through the specification of a SELECT-command as a condition for the UNLOAD, to describe the needed JOIN-function. Finally this unloaded file will serve as the input for uploading the final table. At a first glance, this can look a long and cumbersome process, but first of all it is far quicker than writing a program, and it can be made even easier through the use of ISQL-command-files.

Some practical hints for using LOAD/UNLOAD:

- 1- Create indexes on tables after loading the tables with data, especially when many rows have to be loaded. Creating the indexes afterwards will be faster most of the time.  
One exception of course are *clustering* indexes, which will indeed cluster data rows together based on the similarity of the key-values.
- 2- Avoid the "*subset*"-feature of the ISQL-load-command. When you need only parts of the input-file, process the file with FCOPY's subset option instead. I had an example where FCOPY was 5 times faster than the LOAD-with-subset.

When we consider the conversion of an Image-application to HPSQL, we may not disregard the differences in the program design itself. The following section will discuss this matter in detail.

SQL-access is currently supported for two languages: COBOL and PASCAL. Designing a program that will operate on an HPSQL data base, requires from the programmer a different approach. Especially experienced Image-users will suffer from this in the beginning.

The unit of work in HPSQL is not the logical data base, but a so called Data Base Environment (DBE): the collection of logical data bases sharing the same physical environment, i.e. configuration, disc space management, logfiles, security settings, etc..

An ISQL-session or program with embedded SQL, can only "connect" to one DBE. This has a significant impact on the structure of the application: the source of information has to be centralized in one DBE. If many users will operate simultaneously on the same DBE, special care has to be taken in the design of logical transactions: a transaction is every data base operation between a **BEGIN WORK** and **COMMIT WORK**, to compare with the **DBBEGIN** and **DBEND** calls in Image. As long as a transaction is not committed, modifications are not made permanent (i.e. not accessible for other users), locks on pages are not released and bufferspace is not freed. So, unnecessary long transactions can lockout other users from access to the data base. Every logical transaction, thus, must be designed carefully.

Another peculiarity of the HPSQL-transaction management is the automatic rollback of transactions that will cause a deadlock. Consequently it will be indispensable for application programs to test the condition codes returned by HPSQL in the **SQLCA**-area, for a deadlock-condition, and to allow for resubmission of the transaction.

Speaking about condition codes: there are two ways of handling error conditions. Like in Image, we can test condition codes, retrieve the corresponding error message with **SQLXPLAIN**, and take an appropriate action.

But we can also choose for the automatic exception processing. With the **WHENEVER-CLAUSE**, a programmer can provide labels of procedures that will handle the error-conditions. This can seem very handy, but... be aware that HPSQL branches to those error routines with a "GOTO", and thus there is no easy way of returning to the normal program logic.

There are two types of access in HPSQL that have no direct counterpart in Image: the so called **bulk operations**, and the use of **cursor**s. Bulk operations allow to retrieve with one **SELECT**-command for instance, a specified number of rows. A bulk select can be usefull to find all rows that match a condition, at one stroke, provided you know approximately the number of rows that can satisfy the condition.


The use of cursors permits also to retrieve all rows that match a condition. The mechanism is the following: when you declare a cursor, you specify a cursorname and relate it to a **SELECT**-command. Opening the cursor will execute the select-command, without passing information to the application. The **FETCH** command passes the selected rows one by one to the program. So far we can find some analogy with Image in this description: a **DBFIND** points us to the desired information in a detail data-set, and subsequent **DBGET**'s (mode 5) will pass the satisfying records one by one, till the end of the chain.

But the comparison ends here, for in HPSQL we can have multiple cursors open concurrently, even on the same tables. For instance, to scan through an hierarchical structure, this is the ideal approach [see figure 2]. In Image, one can only force two simultaneous access paths to a detail set by opening the data base twice.

Another interesting SQL-feature is the *dynamic command processing*. The **PREPARE**-command operates on an inputstring, containing the command-image, performs the command parsing and syntax checking and prepares it for execution. In case of a "non-select" statement, **EXECUTE** will perform the requested operation. Dynamic select statements must be translated, with the help of the **DESCRIBE**-command, to the **CURSOR**-mode instruction sequence explained before.

This approach allows, for instance, to use HPSQL in languages that do not support the relational DBMS directly. It is indeed well conceivable to design some general purpose SQL-procedures, i.e. written in Pascal, which are stored in a SL and can be called by mainprograms in Fortran, Transact-SQL, etc..

There are elements, though, to be aware of: HPSQL initializes a SQL communication area, called **SQLCA**, when you connect to a data base environment, and uses this **SQLCA** for all subsequent calls. This area must thus be declared globally in the main program, and passed as a parameter to the SL-procedures. [see figure 3].



<i>HP 3000</i>	<i>DB11/5</i>
<i>INTERNATIONAL CONFERENCE</i>	
<i>VIENNA 1987</i>	

Although this method is not recommended for heavy usage, due to the performance impact, it can provide a excellent workaround if one needs occasional access to an HPSQL data base from within, for instance, a Transact application.

It is not my goal to give a complete picture of performance topics of relational data bases. This subject is worth a few presentations on its own.

I prefer to concentrate on a few design concepts which can have a significant performance impact. First of all, when creating tables, one can define variable length rows by allowing NULL values for one or more columns, or by defining columns as VARCHAR. This method can save a substantial amount of disc space, but can have a negative influence on the performance : when frequent additions and deletions are performed on such a table, this will result in a more intensive free space management for the disc pages (located in DBEFILES) that contain this table. When rows or tuples can vary in length, deleting rows will result in "holes" of varying size. This will often cause tuple-migration within the physical page.

A second topic is the use of the UPDATE STATISTICS command. Each DBE contains as part of the system catalog information on data base usage and space occupation. Executing the UPDATE STATISTICS command for a table will update this information. The COBOL - and PASCAL-preprocessor will use this information to optimize the SQL-statements embedded in the programs.

Executing a sequence of UPDATE STATISTICS-commands is necessary to allow the preprocessors to perform their optimization task. It is also indispensable for allowing the data base administrator to manage the disc space requirements of his data bases.

Indeed, some of the so called System Views, i.e. System.Table, System.Index \_ contain informations on the number of disc-pages used, number of free pages left, etc..This information will only reflect the current status after issuing a Update Statistics command.

## CONCLUSION

This article has no pretension to be a complete guide for the novice HPSQL user. Don't forget that only the Data Base Administration training already takes five days! The main goal was to give practical guidelines, to compare the application design between Image and HPSQL, and last but not least to emphasize the broad range of possibilities of this data base management system. The important message is : get started en learn.

Andre VAN AKEN

*Hewlett Packard Belgium*

# FIG 1: ISQL UNLOAD EXAMPLE TO COMBINE MULTIPLE TABLES

DB11/7

APPENDIX

A

TEMPTABLE1

|emplnr|emplname |

TEMPTABLE2

|emplnr|deptnr|phonenumber |

```
UNLOAD TO INTERNAL unldfile FROM
"SELECT temptable1.emplnr, emplname, deptnr, phonenumber
FROM temptable1, temptable2
WHERE temptable1.emplnr = temptable2.emplnr";
```

-----

LOAD FROM INTERNAL unldfile TO finaltable;

FINALTABLE

|emplnr|emplname |deptnr|phonenumber |

## FIG2: SAMPLE PROGRAM USING MULTIPLE CURSORS

DB11/8

APPENDIX

B

```
$Heap_Dispose ON$
$Heap_Compact ON$
$Uslnit$
$Standard_Level 'HP3000'$
(*****
(* This program produces a structured report of all the *)
(* employees in a departement, using a technique with *)
(* multiple Sql_cursors *)
(*-----*)
(* written by Andre VAN AKEN *)
(* HP Belgium *)
(*****)

Program REPORT(input,output);

const
 OK = 0;
 NotFound = 100;
 DeadLock = -14024;

var
(* ==> Begin Host Variable Declaration *)
EXEC SQL Begin Declare Section;
 EmplNumber : packed array[1..4] of char;
 Emplname : packed array[1..30] of char;
 DeptNumber : packed array[1..4] of char;
 FuncNumber : packed array[1..4] of char;
 SQLmessage : packed array[1..132] of char;
 MgrInd : SqlInd;
 InputDept : packed array[1..4] of char;
 DeptMgr : packed array[1..4] of char;
 DistrictMgr : packed array[1..4] of char;
EXEC SQL End Declare Section;

 SQLCA : SQLCA_type; (* SQL comm.area *)
 Abort : boolean;
 Response : packed array[1..4] of char;
 indent : smallint;

$Page$
procedure SQLStatusCheck; (* Display error messages *)
begin

abort:= FALSE;
if SQLCA.SQLCODE < Deadlock then Abort:= TRUE;
repeat
EXEC SQL SQLEXPLAIN :SQLMessage;
```

Figure 2

```

writeln(SQLMessage);
until SQLCA.SQLCODE = 0;

if Abort then
 begin
 EXEC SQL COMMIT WORK RELEASE;
 halt;
 end;

end; (* end procedure *)

function ConnectDBE: boolean; (* connect to DEMODBE.AVA.HPDB *)
begin

writeln('Please wait for Connect to DEMODBE DataBase Environment');
EXEC SQL CONNECT TO 'DemoDBE.AVA.HPDB';

ConnectDBE := TRUE;
if SQLCA.SQLCODE <> OK then
 begin

 ConnectDBE:= FALSE;
 SQLStatusCheck;

 end;
end; (* end of function *)

procedure ReleaseDBE; (* Release from DemoDBE *)
begin

writeln('Releasing connection from DemoDBE');
EXEC SQL RELEASE;
if SQLCA.SQLCODE <> OK then
 SQLStatusCheck;
end; (* end of procedure *)

$PAGE$
function BeginTransaction: boolean;
begin

EXEC SQL BEGIN WORK;
if SQLCA.SQLCODE <> OK then
 begin
 BeginTransaction:= FALSE;
 SQLStatusCheck;
 end
else
 BeginTransaction:= TRUE;

end; (* end of function *)

procedure EndTransaction;
begin

```



Figure 2

```

EXEC SQL COMMIT WORK;
if SQLCA.SQLCODE <> OK then SQLStatusCheck;

end; (* end of procedure *)

$PAGES$

procedure DisplayHeader;
begin
writeln('Employee List of Departement ', DeptNumber);
writeln('-----');
writeln;
writeln('EMPL | EMPL.NAME | FUNCTION');
writeln;
end; (* end of procedure *)

procedure DisplayRow;
begin

writeln(' ':indent,
 EmplNumber, ' ', EmplName, ' ', FuncNumber);
end;

procedure DeclareCursors;
begin

EXEC SQL DECLARE DEPT_CURSOR CURSOR FOR
 SELECT Emplnr, Emplname, Funcnr
 FROM Persdb.Empl
 WHERE Deptnr = :InputDept AND Mgrnr = :DeptMgr;
EXEC SQL DECLARE DISTRICT_CURSOR CURSOR FOR
 SELECT Emplnr, Emplname, Funcnr
 FROM Persdb.Empl
 WHERE Deptnr = :InputDept AND Mgrnr = :DistrictMgr;

if SQLCA.SQLCODE <> OK then
begin
 SQLStatusCheck;
 ReleasedBE;
end;
end; (* end of procedure *)

function OpenDeptCursor : boolean;
begin

EXEC SQL OPEN DEPT_CURSOR;
if SQLCA.SQLCODE <> OK then
begin
 OpenDeptCursor:= FALSE;
 SQLStatusCheck;
 ReleasedBE;
end
end

```

Figure 2

```

else
 OpenDeptCursor:= TRUE;

end; (* end of function *)

function CloseDeptCursor: boolean;
begin
EXEC SQL CLOSE DEPT_CURSOR;
if SQLCA.SQLCODE <> OK then
 begin
 CloseDeptCursor:= FALSE;
 SQLStatusCheck;
 ReleaseDBE;
 end
else
 CloseDeptCursor:= TRUE;

end; (* end of function*)

function OpenDistrictCursor : boolean;
begin

EXEC SQL OPEN DISTRICT_CURSOR;
if SQLCA.SQLCODE <> OK then
 begin
 OpenDistrictCursor:= FALSE;
 SQLStatusCheck;
 ReleaseDBE;
 end
else
 OpenDistrictCursor:= TRUE;

end; (* end of function *)

function CloseDistrictCursor: boolean;
begin
EXEC SQL CLOSE DISTRICT_CURSOR;
if SQLCA.SQLCODE <> OK then
 begin
 CloseDistrictCursor:= FALSE;
 SQLStatusCheck;
 ReleaseDBE;
 end
else
 CloseDistrictCursor:= TRUE;

end; (* end of function*)

procedure FetchEmpl; (* get employees of 1 district *)
begin

if OpenDistrictCursor then
begin
 while SQLCA.SQLCODE = OK do

```

Figure 2

```

begin
 EXEC SQL FETCH District_Cursor
 INTO :EmplNumber, :EmplName, :FuncNumber ;
 case SQLCA.SQLCODE of
 OK : begin
 indent:= 6;
 DisplayRow;
 end;
 NOTFOUND : begin
 end;
 otherwise SQLStatusCheck;
 end;
end;
if CloseDistrictCursor then
end;
end; (* end of procedure *)

procedure FetchDistricts;
begin
if OpenDeptCursor then
begin
 while SQLCA.SQLCODE = OK do
 begin
 EXEC SQL FETCH Dept_Cursor
 INTO :EmplNumber, :EmplName, :FuncNumber;
 case SQLCA.SQLCODE of
 OK : begin
 indent:=4;
 DisplayRow;
 DistrictMgr:= EmplNumber;
 FetchEmpl;
 end;
 NOTFOUND : begin
 end;
 otherwise SQLStatusCheck;
 end;
 end;
 if CloseDeptCursor then
end;

end; (* end of procedure *)
(* INSERT FETCH PROCEDURE HERE *)

$PAGE$ (* BEGINNING OF MAIN PROGRAM LOOP *)
Begin

Declarecursors;
if ConnectDBE then
begin
 writeln(' Demonstration Program for Reporting from HPSQL');
 writeln(' =====');
 repeat
 writeln;
 prompt('Enter DeptCode or / to STOP >> ');

```

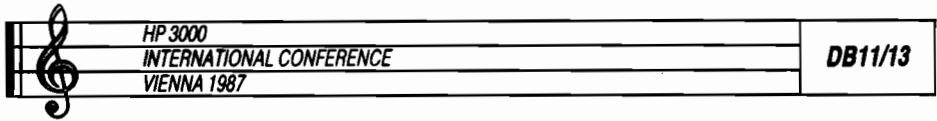


Figure 2

```
readln(Response);
if response[1] <> '/' then
begin
 InputDept:= Response;
 if BeginTransaction then
 begin
 EXEC SQL SELECT Emplnr,Emplname,Funcnr,Mgrnr
 INTO :EmplNumber, :Emplname, :Funcnumber,
 :Deptmgr :MgrInd
 FROM Persdb.Empl
 WHERE Deptnr = :Inputdept AND Mgrnr IS NULL;
 if SQLCA.SQLCODE = OK then
 begin
 DisplayHeader;
 Indent:=2;
 DisplayRow;
 DeptMgr:= EmplNumber;
 FetchDistricts;
 end
 else
 if SQLCA.SQLCODE = NotFound then
 writeln('Head of Departement not in DataBase')
 else
 SQLStatusCheck;
 EndTransaction;
 end
else
 ReleaseDBE;
end;
until Response[1] = '/';
ReleaseDBE;
end
else
 writeln('FAILED to Connect to Demodbe !!!');
end.
```

# FIG3 : SAMPLE PROCEDURES CALLED FROM FOREIGN LANGUAGES

DB11/14

APPENDIX

C

```

$SET 'trace=FALSE'$
$symdebug$
$Check_Formal_Parm 2$
$Subprogram$
$Xref ON$
$Uslnit$
$Standard Level 'HP3000'$
$Segment 'SQLSUB'$
program SqlSub(input,output);

const
 NotFound = 100;
 OK = 0;
 DeadLock = -14024;

 NbrFmtRecords = 64; (* max.number of columns in Select*)
 MaxDataBuff = 3000;

type
 MsgType = packed array[1..132] of char;
 Command_Type = packed array[1..500] of char;
function ConnectDbe(VAR Sqlca: Sqlca_Type;VAR Error:MsgType):boolean;
(*****)
 EXEC SQL BEGIN DECLARE SECTION;
 ConnectError : packed array[1..132] of char;
 EXEC SQL END DECLARE SECTION;
begin
 EXEC SQL CONNECT TO 'dbenvir';
 ConnectDbe:= TRUE;
 if SQLCA.SQLCODE <> OK then
 begin
 ConnectDbe:= FALSE;
 EXEC SQL SQLEXPLAIN :ConnectError;
 Error:= ConnectError;
 writeln('==> ERROR IN CONNECT!');
 end;
end; (* end of ConnectDbe function *)

procedure ReleaseDbe(VAR Sqlca:Sqlca_Type);
(*****)
begin
 EXEC SQL RELEASE;
 if SQLCA.SQLCODE <> OK then
 writeln('==> ERROR IN RELEASE!');
end; (* end of ReleaseDbe procedure *)

function InsertRow(VAR Sqlca:Sqlca_Type;
VAR Command:Command_Type;
VAR Error:Msgtype):boolean;

```

Figure 3

```

(*****)
EXEC SQL BEGIN DECLARE SECTION;
insert_command :packed array[1..500] of char;
insert_error :packed array[1..132] of char;
EXEC SQL END DECLARE SECTION;
begin
 insert_command:= Command;
 InsertRow:= TRUE;
 EXEC SQL BEGIN WORK;
 EXEC SQL EXECUTE IMMEDIATE : Insert_Command;
 if SQLCA.SQLCODE <> OK then
 begin
 EXEC SQL SQLEXPLAIN :insert_error;
 Error:= Insert Error;
 EXEC SQL ROLLBACK WORK;
 Insert_Row:= FALSE;
 end
 else
 begin
 EXEC SQL COMMIT WORK;
 end;
end; (* end of InsertRow procedure *)

begin
(***** dummy outer block *****)
end.

#####
BEGIN
 byte array command(0:499);
 byte array bsqlca(0:499);
 array sqlca(*)=bsqlca;
 byte array error(0:131);
 array text(0:39);
 byte array btext(*) = text;
 logical flag;
logical procedure connectdbe(sqlca, error);
 array sqlca; byte array error;option external;
procedure releasedbe(sqlca);array sqlca;option external;
logical procedure insertrow(sqlca, command, error);
 array sqlca; byte array command, error; option external;
intrinsic read,print;

move btext := "ATTEMPTING connect to 'DEMODOBE' ";
print(text,-32,%40);
flag:= connectdbe(sqlca,error);
if flag = 1 then
 begin
 move btext:= "enter SQL command: ";
 print(text,-19,%320);
 read(command,-500);
 flag:=0;
 flag:=insertrow(sqlca,command,error);
 if flag = 1 then

```

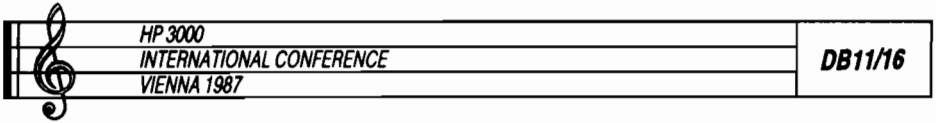


Figure 3

```
begin
move btext:="INSERT command executed succesfully ";
print(text,-36,%40);
end
else
begin
move btext:="INSERT command failed ";
print(text,-22,%40);
move btext:= error, (-80);
print(text,-80,%40);
end;
releasedbe(sqlca);
end
else
begin
move btext:="CONNECT to DEMODBE failed ";
print(text,-25,%40);
end;
end.
```

## Maximizng Your Database Design Through Normalization

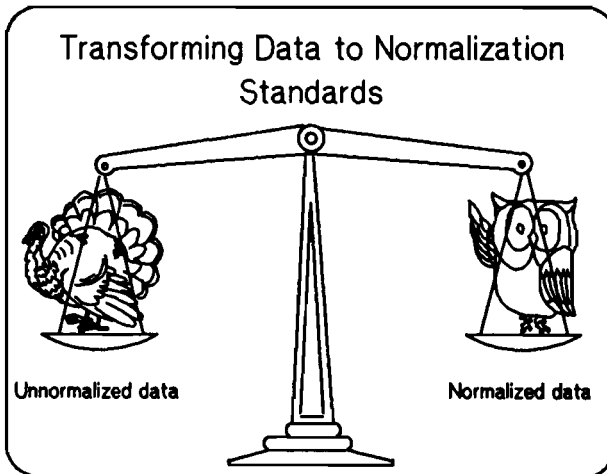
Michele Dingerson

Hewlett Packard Company, Santa Clara, California, USA


### Summary

Normalization is often presented as a complex, mathematically provable approach to designing databases for relational technology. The normalized approach can be used for nonrelational databases as well as relational databases. Moreover, it does not require an understanding of relational theory. This paper presents the guidelines for the normalization without delving into relational theory.

### Introduction





	HP 3000	DB12/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Normalization is the process used to design your database so that performance is maximized and data redundancy is minimized. The primary goal of normalization is to arrange the data so that it can be represented in two-dimensional tabular form.

Normalization rules are designed to prevent data inconsistencies and redundancies. They are, however, biased toward the assumption that all nonkey fields will be updated frequently. The recommendation is to use normalization to put your data into two-dimensional tabular form and then alter the tables to fit the specific requirements of the application.

In the process of normalization, you will usually begin with data that you need to put into tables, not with unnormalized tables. For the sake of illustration, the examples in this paper start with data that is already in an unnormalized table.

The process of normalization involves examining the data to determine if it meets criteria for five normal forms. This paper deals with the first, second, third, and fourth normal forms. It also covers cases where you choose to leave tables in first or second normal form for performance reasons.

This illustration shows an example of unnormalized data. In this case, the record stores information about various projects that an employee has worked on. Note that for each employee there is only one employee name, job code and job title, however, there may be multiple project numbers, completion dates, and hours worked.



### Example of Unnormalized Data

Emp#	Emp Name	Job Code	Job Title	Proj#	Completion Date	Hours Worked

#### First Normal Form

The first step in normalization is to place the data in first normal form. This involves removing all repeating groups of nonkey fields. In this example shown below, repeating groups are those items that appear more than once for a given employee.

Note that this example assumes that each employee has only one job code and title. For example, the same person cannot be a secretary and an engineer.

In the previous example, the repeating group included Proj#, Completion Date, and Hours Worked. Each employee could be working on several projects. Each project would have its own completion date and number of hours worked. To put this data in first normal form, the repeating group is removed and put in a separate table. The two tables that are required to represent the data are shown on the in the illustration below.



### Example of First Normal Form

**Employee**

Emp#	Emp Name	Job Code	Job Title

**TimeAccounting**

Emp#	Proj#	Completion Date	Hours Worked

### Second Normal Form

Second normal form involves the idea of functional dependence. A field is functionally dependent on another field if its value depends on the value of another field. To arrive at second normal form you need to remove any fields that are not fully functionally dependent on the key field.

In this example, it is assumed that there can only be one employee name associated with each employee number. For each employee number there is only one job code and one job title. Emp Name, Job Code, and Job Title are functionally dependent on Emp#.

If it is assumed that employee names are unique, Job Code, Emp# and Job Title are all functionally dependent on Emp Name. Emp# and Emp Name are not, however, dependent on Job Code because more than one employee can have the same job code, and more than one employee may have the same job title. For example, a company may have hundreds of secretaries.

Hours Worked is the total hours worked per employee per project. Hours Worked is dependent on the combination of Emp# and Proj#. If it is assumed that the Completion Date is the date when the entire project is to be completed, Completion Date is dependent only on Proj#. To put the tables in second normal form, Proj# and Completion Date were put in a separate table.

### Example of Second Normal Form

**Employee**

Emp#	Emp Name	Job Code	Job Title

**TimeAccounting**

Emp#	Proj#	Hours Worked

**Projects**

Proj#	Completion Date

### Third Normal Form

Third normal form involves transitive dependence. It requires that every nonkey field is not transitively dependent on the primary key. If transitive dependencies are found, they should be removed to put the tables in third normal form.

If you have three fields, A, B, and C such that C is functionally dependent on B, and B is functionally dependent on A, then C is functionally dependent on A. If A is not functionally dependent on B, or B is not functionally dependent on C, then C is transitively dependent on A.



In this example, a Job Table was made to arrive at third normal form. Before this was done, Job Title was transitively dependent on Emp#. Emp Name, Job Code and Job Title are functionally dependent on Emp#. If it is assumed that there is only one Job Title associated with each Job Code, Job Title is functionally dependent on Job Code. Many employees may have the same Job Code, therefore, Job Title is transitively dependent on Emp#.

### Example of Third Normal Form

Employee

Emp#	Emp Name	Job Code

Job

Job Code	Job Title

TimeAccounting

Emp#	Proj#	Hours Worked

Project

Proj#	Completion Date

Third normal form relieves update problems. For example, in second or first normal form, when a Job Title changed, all records referring to that Job Title would have to change. Because of the redundancy of keeping a Job Title field for each employee number, the data might become inconsistent. In third normal form, only the Job Table would need to be updated.



### Fourth Normal Form

Fourth normal form deals with multivalued fields. A table should not contain two or more independent multivalued fields for a key.

For instance, suppose in the earlier examples that an employee could be assigned to several projects at the same time, and that the employee could have several skills (see the top of the illustration). In this example, you have two many-to-many relationships. To satisfy fourth normal form, these should be split into two tables as shown on the bottom of the illustration.

### Example of Fourth Normal Form

Emp#	Proj#	Skill Code

Fourth Normal Form

EmployeeProject

Emp#	Proj#

EmployeeSkill

Emp#	Skill Code

When fourth normal form is violated, uncertainties in updating arise. For example, one employee might be able to type, take shorthand, and use a dictaphone. That employee might also be working on three different projects. There would be numerous ways to represent these records if fourth normal form was violated. It would be difficult to know what combination was used to represent the data and how that data should be updated if any piece of it changed.



Conclusion

Why Choose Not To Fully Normalize Data

Faster, more efficient retrieval of data

Total Space Required

After your data has been normalized, you should consider unnormalized alternatives based on your applications. This illustration shows two reasons why you might choose to leave your tables in first or second normal forms:

- Retrieval of data may be faster. For example, if all applications were going to retrieve Job Title every time they retrieved Employee Number, Employee Name and Job Code, and never retrieve or update Job Title based on Job Code, it might be more efficient to leave the Employee table in second normal form.
- Total space required may be greater for normalized tables. Therefore, if space is a primary concern, and the normalized tables require more space, the tables should be left in an unnormalized form.

### Bibliography

1. Bass, Paul. "Six Steps to a Normalized Database," *Supergroup Association*, May/June, 1985, pp. 30-38.
2. Codd, C.J. An Introduction to Database Systems. 3rd Ed. Addison Wesley, Reading, MA, 1981.
3. Kent, William. "A Simple Guide to Five Normal Forms in Relational Database Theory," Communications of the ACM, Vol. 26, Number 2, (February, 1983), pp.120-125.
4. Turk, Thomas A. "Using Data Normalization Techniques for Effective Data Base Design," Journal of Information Systems Management, Vol. 2, Number 1,(Winter 1985), pp. 36-49.

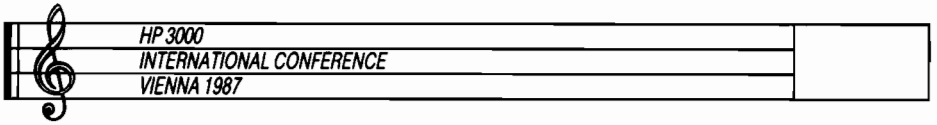
### Biography

Michele Dingerson is currently a Systems Engineer Instructor for Hewlett-Packard. As a full time instructor she teaches various HP3000 Customer Training Courses at the Neely Santa Clara Sales Office.

Prior to her current position, Michele worked as a Marketing Engineer for Hewlett-Packard in the Data Management Support Group of Information Technology Group's Technical Marketing Department. As a member of the documentation and training team for Hewlett-Packard's first relational database product, HPSQL, she developed the first database administration training course ever developed at Hewlett-Packard and participated in training Systems Engineers on database products.

Michele has also worked as a Financial Systems Analyst for a large California-based telecommunications firm. Prior to that position, she worked as a Programmer/Analyst at the Corporate Headquarters of Hewlett-Packard in Palo Alto California. Ms. Dingerson initially joined Hewlett-Packard in 1980, and most recently has been with Hewlett-Packard since 1984.





HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

## Dynamic Aspects of Information Modelling

### Summary:

The Entity-Relationship Model of Mr.Chen is the most important model for information systems today. However that model can only help us to find out the static, non dynamic behaviour of an information system. That paper will show that every information system also has a dynamic behaviour and how to model that dynamic behaviour.

### I. What is an Information System?

A simple answer is: An information system is a picture of the reality or part of the reality of a company or organisation with the help of a computer and its software. The problem is to find out how does a company or organisation or part of a organisation looks like and how does all parts of such a organisation work together?

### II. Modelling

Whenever you try to understand a very complex object you build a model for that subject. That is not only true in physics, civil- or mechanical engineering, but also for information systems. For information systems, the above mentioned Entity-Relationship Model is the standard today. A lot of information-system-design-tools work with that model in mind. Even HP's new SYSTEM-DICTIONARY has integrated that model as a basic working tool. However the dynamic aspects are totally ignored. An important side-effect of the modelling approach for application design is that at least the basics of modelling can be understood by a non technical end-user! The Entity-Relation-Model is a good example for the above statement.

### III. Modelling the dynamic aspects of information systems

#### III.1 Single State Driven

The basic idea is that every entity in an informations system has a so-called state-variable. That state-variable indicates the now active (application) state of that entity. Every entity walks through a lot of states during its life, e.g. for a human bein these state can be: Being a baby, child, pupil, student, single, married, (unfortunately) divorced, (happy) married (again), father or mother, retired, (happy) grandmother or grandfather, died. For the entity Order the states can be: in the state of being entered, to be checked for mistakes, accepted or rejected, in production schedule, produced, delivered, sent to accounting and done. It is the job of the designer of an information system to find out all entities and all relationships of that system and as well as all the states of the entities and, because relationships can also be seen as entities, the states of the relationships to. As a result we get per entity and per relationship a "course of live"! The existence of every entity and every relation-

ship as well as every state of the course of life of every entity/relationship should be accepted and understood by every enduser, otherwise your information system may not be a picture of the reality of your enduser. A entity switches from one state to the next state by the activation and execution of a transaction! As a consequence a totally designed information system consists out of the according entities, the relationships, the course of life with its states and the transactions that actually changes these states. All these subjects must be transparent and understood by the enduser. In practice however, transactions can be very complicated. The above examples are for transactions that are only depending of one single state. (Single state driven). As long as transactions are only depending of one state they are very simple and easy to understand. The problem is more complex for the designer and for the end-user, if transactions are depending of more than one state or preconditions and these transactions switch more than one state or are setting more postconditions. That dynamic behaviour can be defined and modelled by so called PETRI-NETS.

### III.2 PETRI-NETS.

It is not the idea of that paper to discuss in details and mathematically correct all aspects of Petri-Nets. My intentions are two: First to show at some examples how easy complex and dynamic aspects of informations systems can be modelled, and second that the reader becomes motivated to make a deeper look inside of Petri-Nets./1/,/2/ But first a very rough definition of Petri-Nets which is "translated" into the language of an information system designer:

A Petri-Net is a network with:

- conditions (states), represented by ○
- transactions, represented by □
- arrows from preconditions (states) to transactions
- arrows from transactions to postconditions (states)
- markers ● within some states, which indicate start-conditions

In a network out of conditions and transactions

- is a condition a precondition of a transaction if an arrow shows from ○ to □
- is a condition a postcondition of a transaction if an arrow shows from □ to ○
- in every situation a condition can be active or inactive
- a active condition is indicated by a marker
- a transaction can have more than one precondition and more than one postconditions
- a transaction can be executed if all its preconditions are active and all its postconditions are inactive
- after the execution of a transaction all its preconditions are inactive and all its postconditions are active
- a condition can be a precondition for one or more transactions and a postcondition of one or more transactions.



For information system design a certain state of an entity is a condition of a Petri-Net. Not every condition of a Petri-Net has to be a state. Events can be another type of Petri-Net conditions. Figure I shows a not trivial example how Petri-Nets can be used. It represents a small (production) system that consists out of three machines M1, M2 and M3 and two operators O1 and O2. That system can handle orders under following conditions: Every order has to be handled first at M1, afterwards at M2 or M3. Operator O1 is trained for M1 and M2, O2 is trained for M1 and M3. For some Petri-Nets and also for the information system to be modelled its sometimes not easy to set the initial active preconditions correctly for the Petri-Net. In our case in figure I the initial active preconditions are : Order is given, M1 free, O1 and O2 free.

### III.3 Error Handling

The modelling of an application in form of states and networks with conditions can help us to implement a correct error handling mechanism and behaviour of the system. There are two kinds of errorhandling situations:

#### a. Errorhandling by the enduser


Every enduser transaction, which changes the state(s) of a entity(s), should be reversable whenever it makes sense. That possibility allows the enduser to do mistakes, recognize that error and reset at least logically that transaction. With the help of "state driven thinking" and Petri-Nets, it should be an easy task for the designer to implement per transaction a "reverse transaction".

#### b. Errorhandling by the application software

According to our initial definition, a transaction changes the state(s) of a an entity(s) from one consistent state(s) into another consistent state(s). When a programm or even the computer systems aborts, a transaction may be not completed and one or more entities may be in incomplete states. It is the job of the programmer to guarantee with the help of own or standard errorhandling- and recovery software, that logical inconsistencies never can happen. Petri-Nets is a mental tool to reach that goal.

### IV. Conclusion

By the marriage of the Entity-Relationship-Model with Petri-Nets the designer has now a strong mental tool at his disposal that allows him to model and design complex information systems very easy and very fast and flexible for future changes. The modelling with Petri-Nets should be integrated in every modern design tool, why not also in HP's new SYSTEM DICTIONARY!

	HP.3000	DB13/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Literature:

- /1/ Wolfgang Reisig Systementwurf mit Netzen Springer Verlag Heidelberg, Berlin 1985
- /2/ J. L. Peterson Petri Net Theory and the Modelling of Systems Prentice Hall Inc., Engelwood Cliffs, N. J. 07632, 1981
- /3/ P. P. Chen The Entity-Relationship Model - Toward a Unified View of Data.  
ACM ToDS, Vol.1, No.1 (March 76), pp.9-36

About the Author

Ewald Maria Mund works as an independant consultant. His main interests are design and implementation of information systems as well as fourth generation languages.

Address:

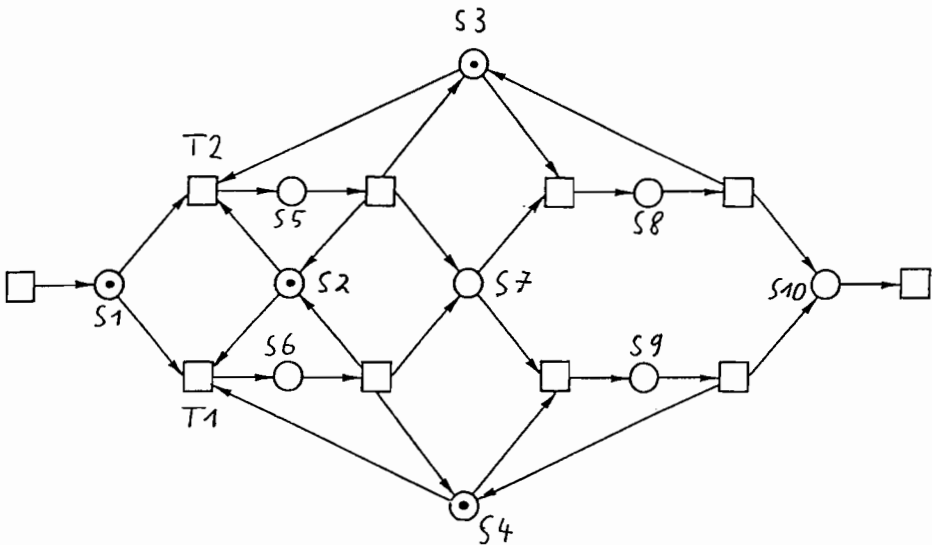
Ewald M. Mund  
Buchholzstr. 13



CH-3066 Stettlen/Bern  
Switzerland

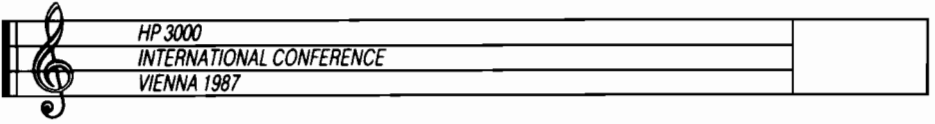
States (S) and Transactions (T) of figure I

- S1 = Order is existing (initial state)
  - S2 = M2 free (initial state)
  - S3 = O1 free (initial state)
  - S4 = O2 free (initial state)
  - S5 = O1 working at M1
  - S6 = O2 working at M2
  - S7 = Order finished at M1
  - S8 = O1 working at M2
  - S9 = O2 working at M3
  - S10 = Order finished
- 
- T1 = O2 accepts order
  - T2 = O1 accepts order

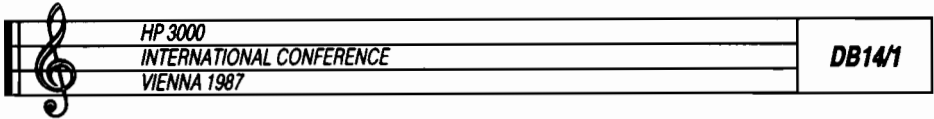


Address: Ewald Maria Mund  
Buchholzstrasse 13  
CH-3066 STETTLEN/BERN  
Switzerland

Tel: 031/519839



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987



## Million Record Database Strategies

By

Cliff Lazar


President  
SystemsExpress  
15015 Ventura Blvd.  
Sherman Oaks, CA 91403  
818/784-6966

### Table of Contents

Introduction
Evolutionary Performance Degradation
Database Mental Retardation
Mice are Different than Elephants
Responsiveness or Dr. Ruth's Good and Bad Databases
Access Strategies
HP Manual Way
Programmer Intensive
Partitioned Datasets
The Entity Detail
Blocking Factors
Backup Strategies
Archive Strategies -- the WORM Drive
Read-only Datasets
Dormant On-site Compressed Data
Dormant Off-line, On-site Compressed Data
Conclusions

Warning: This article is more theoretical than empirical and so the ideas set forth herein need to be tested in each environment before any implementation.



	HP 3000	<b>DB14/2</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Introduction

Million record databases are the justification for many of the HP sites in our community. Million record databases are also the source of many of the problems for those HP sites.

For our purposes, a million record database can be comprised of a few datasets with over 100,000 records each or a massive dataset with over a million records. Such a single massive set is unlikely because it would tend to fall of its own weight without support from side index files.

If there are any who will volunteer that they have such a massive stand-alone file, we should all have sympathy for them.

Some large databases start off strutting with good performance and then evolve into stumbling poor performers. Others, because of poor design, start off poorly and just get worse. In this article I will look at some of the causes of poor database performance on the megabase level and suggest some strategies for improving performance.

In general, the strategies that work for million record, seven figure databases should also apply to six figure and to a degree, even to high four figure databases. Three figure databases, with capacities under 1000, live within the power of the computer, and poor design is compensated by expensive hardware.


## Evolutionary Performance Degradation

In some cases the bases grow from modest beginnings and as a result grow to become inefficient and counter-productive in an evolutionary manner. An observation of psychologists is that if a stimulus changes less than 2% of the base amount, most humans will not be able to tell the difference. Another observation is that if system performances are jumping up and down around a trend, most humans will not be able to discern the trend without keeping records and computing running averages.

Chronic gamblers suffer from fixating on episodic reinforcement; chronic optimists never forget the times the system gives them three second response time and overlook average five minute waits as aberrations. Serial searches in hashed masters will, by the nature of the hash, episodically give very quick responses.

Thus, when a database, with a poor design, begins its life with little data and few users, its performance is very good. Most retrievals occur at finger speed and batch jobs consume very little CPU and even wall clock time. As the base grows and the capacities are reset to higher figures things begin to slow down a little bit at a time.

If the transaction rates are relatively constant, such as sales orders or work hours, the actual percentage increases in database size will decrease each period. Thus, when things begin to

	HP 3000	DB14/3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

suffer the percentage increases in response time may be below the threshold of perception.

Parallel with creeping degradation, typically, the staffing of the data entry and reporting functions are increasing and being trained, so there is little room for comparison based on extended observation.

#### **Database Mental Retardation**

When a baby is born the general expectation is that it will develop into a normally intelligent human with reasonable response time. During the infancy we don't expect the baby to act like a normal adult.

During childhood we watch for signs of improvement. During adolescence we tolerate accidents and lapses because we love our offspring but at some point, usually during childhood, if we don't see improvement and a trend toward normal intelligence we may be forced to admit that our child has chronic problem that won't be solved with age.

Some databases are that way. Their physical structure is such that they will never have normally intelligent response times. Databases with capacities under 1000 entries, while having chronic physical problems, don't exhibit them because their size can be overwhelmed by the computer's speed.


Million record databases with chronic physical problems will be irretrievably inefficient. Adding a disk drive won't help. Going from one MIP to four MIPS won't help. Cacheing will contribute little or no relief. The design is not only inefficient, it is big and inefficient.

#### **Mice are Different than Elephants**

Mice and Elephants are both mammals, with common characteristics like vertebrae, hair, breasts and well-developed brains. At the same time they are very different because of their size. In some ways the elephants suffer from diseconomies of scale. Feeding them becomes a major logistical problem. Elephants can practically deforest an area as they feed. They endanger other nearby animals. Cleaning up after them is a significant issue if you own some. They require alot of real estate, exponentially more, not arithmetically more than mice.

As with mega databases, Elephants require greater strength in their structure. Their muscles are a higher percentage of slow twitch--they can't react rapidly. Their nervous systems require greater decentralization because the signals have a greater distance to travel. They require specially trained handlers.

You can have hundreds of mice in your barn yard and not notice them, but one elephant will make a big impression.

	HP 3000	<b>DB14/4</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Million record databases have similar diseconomies of scale. The obvious issues are

- o Floor space for the disk drives
- o Response time for retrievals
- o Backup time
- o Integrity of the data
- o Dedicated skill levels needed for maintenance

HP and other manufacturer's are offering a solution to the floor space problem with the 4 gigabytes that will squeeze into the same six square feet as an HP7933 disk drive. This still ignores, or even contributes to back-up and archive problems.

#### **Responsiveness or Dr. Ruth's Good and Bad Databases**

A good database is one that will allow the users to retrieve their data when they want it.

Conversely a bad database will not yield data as fast as the user wants and often not as fast as a manual alternative.

The goodness and badness of a database is a function of its size more than its structure. In the case of million record databases this becomes very apparent.

Consider a box of 10 driver's licenses. You can spread them out on a table and get whatever information you want instantly. You don't need Image. You don't even need a computer! But what if you have 10,000 driver's licenses in the same box. Now you can't find anything you want.

How will you sort your driver's licenses? By the meaningless driver's license number (which is the equivalent of the Manual Master Unique key), by last name, by city, by sex? Any sort will obviate any other sort, if only one sort is possible as in the case of an Image manual master. What is the solution?

For decades, libraries have maintained multiple copies of their cards: by author, subject and title. You can go into a library with a million books and find the reference card you want in less than two minutes because you do a random access search on a sorted key and not a serial search.

A million record master on an HP 3000/68 will take 5.8 hours for the average stand-alone serial search. In a multi-user environment every other process will suffer, and the search will take longer.

If the dataset only has fifty to a hundred customer records then retrieval takes place at finger speed and it's O.K. to have a manual master with serial search.

So let's draw an inference. Small masters are O.K. Large masters, with over 1000 entries, make bad databases. A million record manual master makes an abomination.

Next time someone wants to sell you a package, don't accept a demonstration based on less than 100,000 records, if that is going to be your environment.

Anyone who buys a database system which will become 400 byte wide masters with 100,000 to 1,000,000+ capacity should have his head examined. You ignore diseconomies of scale at your peril. You should always get a performance guarantee for the likely range of set capacities your shop will use.

### Physical Realities


A million record database has certain physical realities that are inherent in its size. At 400 bytes wide, a million records will contain 400 million bytes. If the data is stored in a master then 1.25 million records will be needed to avoid the migrating secondaries. That means that the disk drives will need 500 million bytes. Until now that required two 7933 drives and at least one GIC.

### Access Strategies

#### Four Possible Approaches

There are four possible approaches to the design of a customer information file:

- o The HP Manual Way - put the customer data in a master file
- o The programmer intensive Way -- Keep the information in the master but build a set of key index files to compensate for the unsearchability of the master. Many programmers have homegrown approaches to this and Omnidex offers a canned IMSAM approach for about \$10,000.
- o Partitioned Datasets -- The data are partitioned into groupings that vary from high probability of hits to very low probability. The high probability set is searched first. If it is small enough it may reside in cache memory most of the time and the responses may be at CPU speed.
- o The Entity Detail -- store the customer information in a detail but assure that the entries are unique. Systems-Express, our company, offers a retrofit system to copy the data from masters to details and provide rapid search with a combination of automatic masters and a KSAM pointer file, if generic search is desired. The cost varies from \$3,200 to \$6,000, if Cobol source is desired. The approach can be used to redesign an existing database or make easy-access extract databases for inquiries and adhoc reports.

	HP 3000	<b>DB14/6</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

**The HP Manual Way - put the customer data in a master file**

I feel I have to appologize to the HP community. Early on, I wrote a beginner's guide to building Image databases. It was based on the HP Image manual and my readings in the various proceedings. The basic approach was that entities should be stored in Image masters and transactions should be stored in details.

The problem arises when the user wants to extract data not based on the meaningless unique key of the master but on some memorable item such as a company name, or a contact name, or a city, or even a phone number.

I have spoken to numerous users who have bought or built megabases with 300,000 record masters with no pointers except the meaningless key. The key must be meaningless or it couldn't be unique, if there are over 100,000 of them. Exception: part numbers with some rational naming convention. These users tell me that if they want non-key retrievals they must set up stream jobs to run over night or over the week-end.

Other users tell me that they make their living just building duplicative extract files so that the users can get the data they need when they need it.

Let me recant: Don't build databases with the entities in the manual masters. The only thing that Masters are good for is spell-checking.


Put entity data in details and point to them with automatic masters or KSAM files or MPE side files or IMSAM files. More on details below.

**Programmer Intensive--Keep the information in the master,  
but build pointer files**

The operational word is "keep." The assumption is that you inherited the database from someone who was fired or was promoted before management understood the problem.

For technical or political reasons you can't change the structure of the database and convert the masters to details. The solution to getting improved performance is to create side index files which point to the key in the master. KSAM, MPE and IMSAM are good approaches. The appeal of KSAM is that it comes free with HP 3000's and you don't have to build search software.

The approach we follow is fill the KSAM file with the correct spelling of the key values, and not record numbers that are subject to rapid change. This will help to avoid maintaining the KSAM file every time a record is deleted.

	HP 3000	DB147
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

**Partitioned Datasets--** The data are partitioned into groupings that vary from high probability of hits to very low probability

Consider a transaction detail with one million records, which describe events that take place over time. While it may be advantageous to keep all the records online, it may be possible to partition the detail into a few datasets, one or two of which have high probabilities of search hits:

- o Current Day's input
- o Latest Week
- o Latest Month
- o Residual of the data

The Current Day's and Latest Week may have the highest probability of search hits and also will be the smaller of the files. With large blocking factors and a big cache much of the high probability files could be RAM resident most of the time.

At the end of each of the data collection periods the data is posted to the next more comprehensive dataset and deleted from its original source.

The smaller files also have the advantage that if there is a system failure, less is involved in reconstructing the smaller files.

**The Entity Detail --** Store the customer information in a detail but assure that the entries are unique

An entity detail is a detail with one or more keys that are pointed to with automatic masters where at least one key is limited to a chain length of one. The result is that the key is unique, just like a master. The advantage is that there can be as many as 16 keys and that the detail itself does not require excess space for migrating secondaries. The automatic masters do.

Typically a detail may have three to seven keys and typically only one will be unique.

We add a KSAM pointer file for all the keys that we would like generic search. Generally only one to three of the keys will be designated as generic search keys. The KSAM file will contain as many records as the combined capacities of the automatic masters and the manual masters that are to be pointed to, and the record size is equal to the largest field to be pointed to. Thus, this side index file makes trading space for time a very significant tradeoff. At the same time many of our users appreciate the ease of retrieval that generic search offers to the online user.

The typical response time for a large database with a KSAM front-end generic search is less than one second. There is no noticeable difference for a direct key search, with the full value spelling, bypassing the KSAM look-up.

In Table 1 are displayed the theoretical space requirements and search times for a standard manual master and an entity detail. The Image implicit pointer bytes have been ignored. The search time estimates include the time to serial search through empty records in the master. The searches for data in the details assumes that keys are available in automatic masters.

Table 1 - Theoretical Space Requirements for 1,000,000  
Customer Records and Search Times  
(MM=000,000)  
(400 byte wide record with 100 bytes of keys)

Structure	Bytes in Bytes		Bytes in		Total Search Time
	Keys	Detail	KSAM	File	
	(1,2)		(3)		(5)
	---	---	---	---	---
Standard All Manual	1 500MM	0	0	500MM	5.8 hours
Detail with 5 5 automatics	5 125MM	400MM	0	525MM	1 second
Detail with 5 5 automatics 1 KSAM with 5 pointers	5 125MM	400MM	125MM	650MM	<1 second

Footnotes:


- (1) 400 bytes \* 1MM \* 1.25 = 500MM for migrating secondaries avoidance
- (2) automatic masters = 100 bytes \* 1MM \* 1.25 = 125MM  
Some of the automatics may be much less than 1MM
- (3) 20 bytes \* 5 \* 1MM \* 1.25 = 125MM
- (4) Sum of Bytes in Masters, details and KSAM files
- (5) 500MM/30 accesses per second/3600 seconds per hour/2  
This can be substantially improved through better blocking factors  
< 1 second adjusts for the savings in user keystrokes with generic search

Entity details are used by some shops as extract databases for quick user access to the data that is otherwise locked in ugly masters. :DBTRANS or Supertool is used to copy the data and :DBEXPRESS or grunt Cobol is used to build the retrieval system.

Blocking Factors

Well selected blocking factors can substantially improve the efficiency of database retrievals. The more records that are retrieved with each seek, the less time and resouce consuming seeks are necessary.

The common wisdom has been that 25% of a master's space should be empty space to avoid migrating secondaries outside of the block. It has been suggested (Van Valkenburgh, Interex 86, Detroit Paper 3105, p.23) that the free space can be limited to the reciprocal

	HP 3000	DB14/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

of the blocking factor. I have run a simple simulation of this approach and it appears that for high blocking factors, the likelihood of not finding free space in any given block is high enough to expect a lot of migrating secondaries to cross over the block boundaries if the free space is limited to the reciprocal.

This is another good reason to avoid using masters except for spell checkers.

### **Backup Strategies**

Million record databases require lots of backup. While backing up only changed datasets is attractive there may be a problem with partial database backups that can jeopardize database integrity. It may be that keeping the high probability smaller datasets in a separate database can allow nightly backup of only the smaller sets. This can save substantial operator time.

### **Archive Strategies -- the WORM Drive**

Currently archiving million byte databases poses a dual problem of database security and timely archive access.

Security considerations require that backups be kept off site to prevent catastrophic loss and unintentional operator destruction of what look like convenient scratch tapes. Million record databases require large tape storage. Now is the time to consider adopting the write once read many (WORM) optical storage.


WORM drives, shown at COMDEX have capacities in the multiple trillions of bytes. They can deliver any record within 30 seconds and occupy eight square feet of floor space.

Two copies of archival data can be made, with one stored off-site and the other stored conveniently on-site and possibly even on-line in an archival database. The archival database need be opened only when the older data was required.

Implementation of WORM technology will change current database practices in many HP shops. Much of the data currently kept on disks has little current access, but is stored in the database because archiving is so inconvenient if the data is unexpectedly needed. It is another example of episodic reinforcement. The one time in two years when old data was needed, it took hours or days to get it back on the machine so the Database Administrator decided to keep as much data on-line as there was floor space available.

Duplicate WORM optical storage provides the dual advantages of security from loss and accidental scratch over-writes and rapid access to historic data. Experience suggests that the access programs, along with documentation, be stored with the data. It is likely that the programs will change over time and current programs will be unable to read the old data.



	HP 3000	<b>DB14/10</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Hewlett-Packard is currently examining the WORM technology and may have a product on the market in the near future. Other vendors have WORM systems available now that can be interfaced to the HP 3000.

### **Read-only Datasets**

An intermediate solution to the current non-existence of the WORM and the desire to keep megabytes of data online but with smaller disk impact is the concept of a "Read-only Dataset."

A read-only dataset is compressed into 25% or less space that an uncompressed dataset but can only be read from and not written to. Much of the data that we keep online in read/write datasets is, in fact, read-only data. It is historical--employee work hours or customer payments that we want to read but also don't want to over write.

One college in California keeps all its student records online since 1929. They are never written to, only read from, after the close of the school year. Those records occupy eight 7933H disk drives, about four gigabytes. It's all read-only data.

There are other examples of read-only data:

- o Technical Abstracts
  - o Property Records
  - o Library Book References
  - o Stock Transactions
  - o Chemical Formulas and Mathematical Equations
  - o Census Data
  - o Voting Patterns
  - o Telemetry/Sensor Readings

This class of data can be called compressed but read-only.

### **Dormant On-line Compressed Data**

It is also possible to compress data and store it on-line in a dormant state and then decompress it when it is needed. This has the advantage that it is available within minutes but it is not readable or writeable. Such data can be kept online while the image of it is archived at a remote location. The cost is relatively low to have this option. My company offers such a compression/decompression tool. It's called :COMPFILE. You can have your data both ways--archived and available but out of the way.

### **Dormant Off-line, On-site Compressed Data**

The offline counterpart of WORM is highly compressed data on tape. The advantage is that you can save tape, save backup time and you can keep potentially need copies of the data onsite. We of course offer :COMPTAPE for this function.

## Conclusions

Up to the current time many million record databases have suffered from diseconomies of scale. They are so large and so poorly designed that they perform poorly. Maintaining their security and integrity has resulted in a bureaucratic environment where response time has suffered and improvements were avoided.

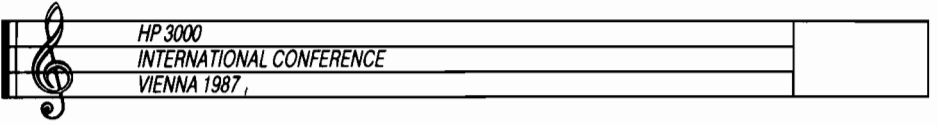
Technology available now can allow the users of million record databases to gain more rapid access to the data through use of entity details or pointer files or extract files. The response time improvements for unkeyed versus keyed access is thousands of seconds versus sub-second response.

Switching from a master to a detail with automatic masters achieves the sub-second response time at the cost of increasing the database size by about 5% to 25%. Adding a generic search KSAM can double the data space requirements, but makes access more user-sympathetic.

Shifting from tape archiving to WORM optical can increase data security while improving data accessibility.

Exploiting data compression techniques promises to give the user site more rapid access to data, with increased security while achieving substantial savings in disk space.

While the majority of the databases are small and reasonably efficient, my observations indicate that the majority of the disk space in the HP community is occupied by large, inefficient databases that can be improved with reasonable effort.



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

EXPERIENCE WITH AN IEEE 802.3 LOCAL AREA NETWORK IN A  
MULTI-VENDOR ENVIRONMENT

W. Schmatz, MAN Technologie GmbH, München

ABSTRACT

The local area network (LAN) techniques may be a strategic approach to some fundamental information processing needs. The MAN Technologie GmbH started the implementation of a local area network for

- \_ integration of applications
- \_ access to central processing

The company's specific situation with data processing equipment of different vendors led to a standard IEEE 802.3 solution with multistar topology on the basis of fibre optic cabling.


From a manager's viewpoint the following topics are discussed:

- \_ availability of LAN-products
- \_ functionality
- \_ throughput
- \_ maintainability
- \_ system price

The result may encourage potential new users.

COPYRIGHT

Parts of this paper were presented on the COMPAUTO 87 International Conference, Geneva, 10-12 March 1987, organized by the Computational Mechanics Institute, Southampton, UK.

	HP 3000	<b>DC01/2</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## STARTING POSITION

### Company's profile

MAN Technologie is the research and development centre of the MAN group. Here some 1000 engineers and scientists are working on new products, future-oriented projects and state-of-the art technologies.

The company is a leading partner in the European aerospace program ARIANE, with additional developments in satellite engineering. Other products concern wind energy plants with a power output from 20 kW to 1.5 MW. Thanks to the expertise in engine technology a prime contribution to the development of cogeneration plants was achieved. Special knowhow is available to produce gas centrifuges for uranium enrichment.

### Information processing applications

A variety of computer assisted tasks run on machines of different vendors. Starting from design, CADAM and CATIA on IBM are used. On a VAX8500 modelling for finite element analysis is undertaken with PATRAN, producing input for NASTRAN and MARC and processing the output for coloured printouts. On the same computer the integrated 2D/CAD-CAM system EQINOX is introduced to generate intermediate drafts of complex mechanical parts and to create interactive the numerical code for NC-machining.

The MAN Technologie operates different minicomputers for 3D-measuring devices and automatized testing facilities.

In the commercial field two HP3000 are used, linked to IBM mainframes.

Most applications have longterm histories and were selected to fit the specific needs best. This situation is often reported. The information processing manager's strategic task for integration is well known.

### Geographic implication

With respect to the theme, the widespread geographic situation is of importance. The laboratories, production floors and offices are spread over 30 buildings. The data processing is in effect distributed by topological reasons. The economic access to central services has to be organized.

## THE LOCAL AREA NETWORK APPROACH

### General LAN-functions

There is no general definition for local area networks. The common understanding is: base for open and universal communication even between different systems/equipments.

The LAN may serve different objectives:

- linking mainframes
- linking minis and/or personal computers
- supporting terminal access
- supporting the base for distributed processing
- supporting manufacturing tasks
- supporting backbone-functions for linking subnetworks

The special suitability of an individual local area network supporting one of the above mentioned characteristic or any combination had to be evaluated.

### Company's objectives in LAN application

The primary intention was to utilize the LAN-technique as a technical approach to some fundamental information processing needs. In a first stage, two goals should be reached.

Integration of applications In a distributed environment you need many services to integrate your application. Leading vendors offer today standard software for the following topics:

- Full network service within one vendor's product line (e.g. within DECNET you get file transfer, remote file/peripheral access, program to program communication)
- Fast file transfer between applications on different computers (e.g. transfer of images from a host to a workstation, specialized for animation or fast transfer of IGES-files for CAD-CAD data exchange)

Access to central processing Traditional terminal-connections resemble a tree; the radix is a single computer. Connecting a device via a server to a local area network may change this structure:

- The logical connection to a series of computers within one homogenous system becomes standard (e.g. a printer server may be attached to a network, to do all printing work for a couple of processors)
- The access of terminals to all computing servers within one homogenous system is standard. Hopefully some terminal server protocols will be enriched by intelligent procedures for access to foreign computers./1/

### Using standards

The commitment for using standards is a good practice. When the MAN Technologie started in 1985 the market leader had not announced his product. Therefore IEEE-standards were investigated.

### THREE STANDARD LOCAL AREA NETWORK TYPES

The following table represents the basic characteristics of three standard local area networks. IEEE 802.3 and its predecessor Ethernet are widely used in thousands of ways around the world. IEEE 802.4, the so called Manufacturing Automation Protocol (MAP), will be operative within the next year. IEEE 802.5 is adopted by IBM as token ring solution. IBM-Products have been available since 1986.

### Access methods

Two very different access methods have been established. CSMA/CD uses a stochastic model:

- a station - ready to send - senses the carrier until the medium is free
- the station starts sending; there is a risk that another station starts within the signal run-time
- collisions must be handled.

Token access methods show a deterministic behaviour by enabling and disabling sending permission.

There is a broad discussion about the advantages/disadvantages of these methods. A comparison is given in /2/.

### ADVANTAGES OF A FIBREOPTIC BASED IEEE 802.3 LAN


#### CSMA/CD-Advantages

Besides the availability, the propagation and magnitude of component-market, one keypoint was decisive. The CSMA/CD-access method permits the coexistence of Ethernet and IEEE 802.3 on the same network. In mid 1986 Hewlett-Packard's software products followed IEEE 802.3, Digital Equipment's products followed Ethernet version 2.

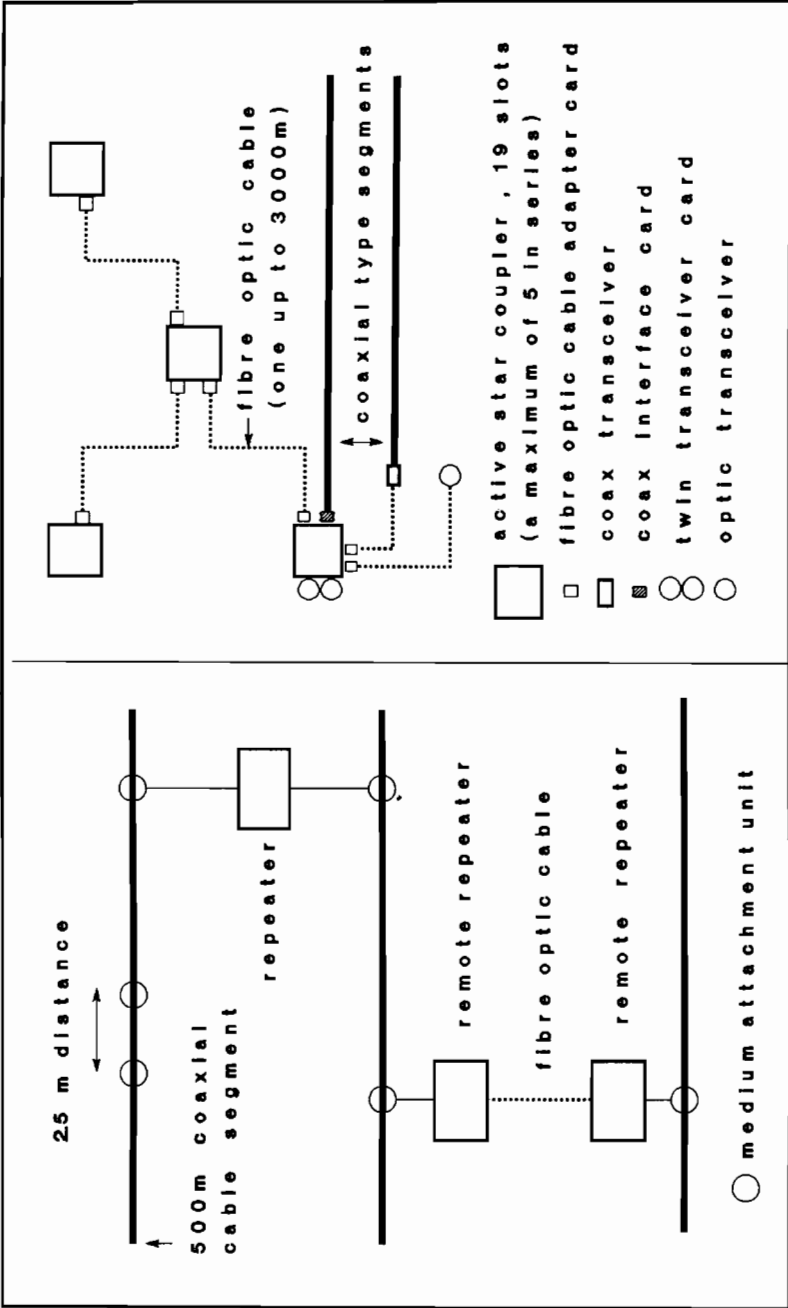
#### Fibreoptic advantages

It was impressing to see, how innovative designers found a way to enlarge the concept of IEEE 802.3. They kept the interface of the medium-attachment-unit exactly as it was. They changed the physical medium completely, enriched the topology and widened the regional application. The following figure 1 shows the conceptual differences.




Standard	Characteristics	Availability
IEEE 802.3	baseband 10 Mbit/sec CSMA/CD bus/multistar	Ethernet ver.2 1982 IEEE 802.3 1985
IEEE 802.4 ( MAP )	broadband 6 MHz token bus	Version 3.0 1988
IEEE 802.5 ( IBM )	baseband 4 Mbits/sec token ring	1986
 <b>MAN</b> Technologie		<b>Standard local area networks</b>





Coax vs. fibre optic LAN-type fig.1



	HP 3000	<b>DC017</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The MAN Technologie decided for the first german supplier, Richard Hirschmann, Radiotechnisches Werk, Esslingen/Neckar, for the following reasons:

No grounding problems The initial network is spanned over two buildings. The coaxial type version demands either for careful grounding or for a fibre optic based remote repeater. Both solutions are expensive. The latter one limits the application, if multiply applicated.

The fibre optic solution is by its nature free of this problem.

Topological flexibility The conventional way, to link all equipment, is suitable for small installations. A better approach is the topology, which IBM has elected for its cabling system. The multistar concept of Hirschmann supports this idea. In every building one star can serve all necessary connections. Additionally transitions to the coaxial type are possible at any star coupler.

Wide range Each segment of the coaxial based type is limited to 500m. The maximum of 2 repeaters limits the range. On the optical medium the deformation of the digital signal is smaller. The range extends to 4500m including one single line of 3000 m and a maximum of five stars in series.

Repeater free base network The absence of repeaters cuts the signal run time. Insofar it reduces collisions by a small amount.

## EXPERIENCE AND PERFORMANCE

### Functionality

We investigated three types of LAN-services:

\_ Type 1: terminal access to central computing. Terminals are hardwired to a terminal server (RS232 up to 19.2 kbaud). The terminal server communicates with the computer via the LAN. Involved products are:

Digital's DECSA-CA, LAT-software, version 2.0.

Digital's VAX8500, operating system VMS, version 4.5, DECnet software (end node license).

To establish the communication, the operator has to download the LAT-software from the host to the terminal server (DECnet-function), which is done by the startup procedure normally. The software spans a fixed relation between all ports of the server and the loading computer.

Type 2: virtual terminal service.

Terminals are hardwired to system A. Computer A and computer B are linked to the LAN. In addition, system B emulates the SNA Physical Unit Type 2, Logical Unit Type 1, 2, 3 devices. The involved products are:

Hewlett Packard's Series HP3000 computers, model /48 and /70, operating system MPE5, UB-Mit, HP AdvanceNet products NS3000/V, LAN/V Link, SNA Link. IBM's mainframe model 3081 running MVS.

To establish the communication, the network software has to be activated on every node. A user starts a session on system A first; issuing the command REMOTE, he will be prompted to start a second session on system B. If the SNA-link is active he may start the SNA-services on the IBM-system. Recently Hewlett Packard offers an enhanced product ACCESS. The computer running SNA Link will be known within the network as SNA-gateway. On demand the connection to that node will be done by ACCESS.

Type 3: file transfer service.

The hardware and software configuration of type 2 is used. For convenience of operating and peripheral sharing the backup of system A is done via the LAN, using a magnetic tape drive, connected to system B.

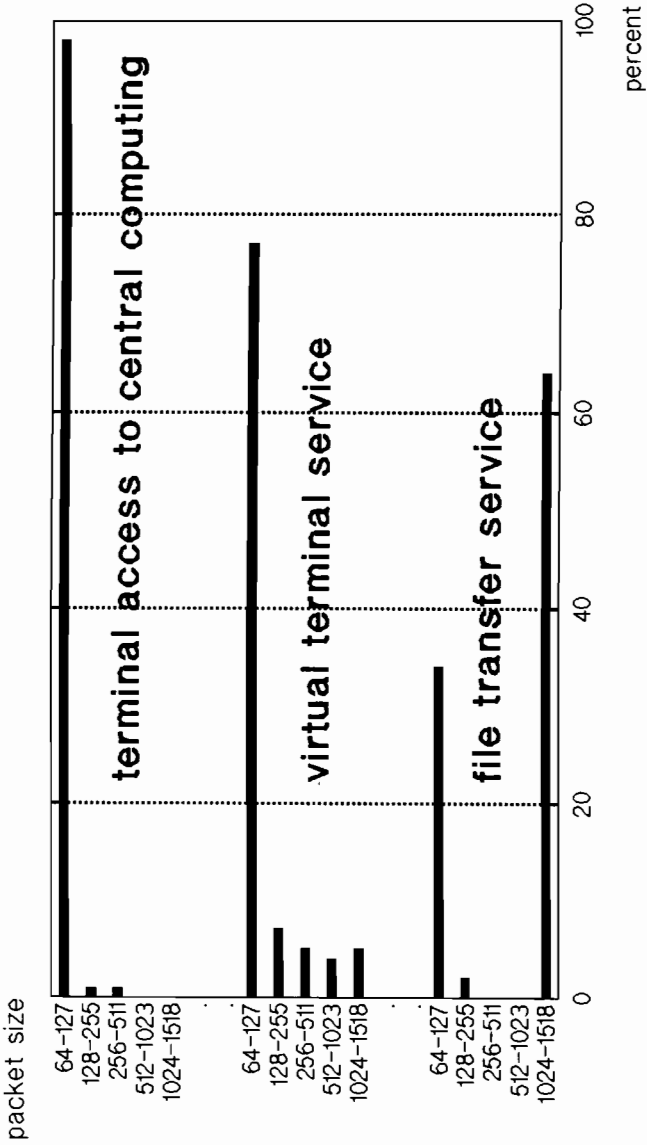
All types of services were installed without major complication. They run concurrently on the same LAN, which proves our assumption of coexistence of Ethernet version 2.0 and IEEE 802.3.

Performance

Most modern communication software provides some tracing facilities. A multivendor network is monitored best with special analysers. We use a EXCELAN/LANalyser, distributed by Synelec Datensysteme GmbH, Munich. It's a PC-based software, working with a special LAN-interface, which fits into a PC-XT, PC-AT or compatible PC's.


Our starting configuration with three CPU's and one terminal server did not produce sufficient load, to beat the throughput limit. Tight analysis of single functions estimates some risks (fig. 2).

With respect to type 1 we found a strange behaviour. If only one character (in character mode) is input, that byte will be enveloped in a 64 byte packet and transferred to the VAX. The acknowledgement (including the echo) produces a further three packets, one of size 64 byte, two of size 68 byte. Terminal server and computer play ping-pong; the overhead is 26 300 %. From another viewpoint: the nominal rate of 10 Mbit/sec will be degraded to 37,9 kbit/sec effective rate.



Packet distributions , 3 types fig.2



	HP 3000	DC01/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Information sent by the computer fills the packets to a limit of 280 bytes. Nevertheless, it is obvious, that character-oriented software, like the VAX-editor EDT does not fit well into the packet-oriented transmission concept.

Type 2 service with screen oriented applications shows a distribution of packet sizes, moving to large packets.

In type 3 service the maximal packet size dominates (it is defined to 1518 bytes). From the operator's view the remote backup has an overhead of 35 % compared with a local backup.

From this very simple analysis it be seen, that software has a tremendous impact on LAN-performance. The software has to bundle as many request as possible on one node to produce large packets.

A theoretical discussion of LAN performance is found in /3/. Large installations are discussed in /4/.

## INSTALLATION AND MAINTAINABILITY

### Fibre optic installation

All installation work was carried out with the company's own staff. A skilled person will learn the mounting of connectors within a day. There is only one limiting factor: for repair of connectors about two hours are needed, mostly for hardening of adhesive.

The transmission should be measured. For small installations a visual inspection is sufficient.

### System installation

In all cases the system services were first proven with the vendors coaxial type. The switching to the fibre optic type was a task of a few minutes. In one case the light emission had to be adjusted; this is possible in three stages by jumpers.

### Maintainability

All cards have LEDs for the following indications: "power", "data accepted" and "collision detection". They are helpful for installation and operator control. For performance measurements and problem solving special LAN-analysers are necessary. Personal computer based analysers are the cheapest solutions.

## SYSTEM PRICE

On the pricelist of the manufacturer mentioned above the following simple configurations are calculated (rounded prices in DM, base 1986):

### Point to point configuration, 300 m distance

- 2 transceiver	5.500 DM
- 300 m fibreoptic cable	3.000 DM
	-----
total	8.500 DM

### Start to star, 4 direct, 4 remote interfaces, 500 m cable

- 2 active star coupler	10.000 DM
- 2 twin transceiver cards	
- fitting into the star coupler	3.000 DM
- 6 fibreoptic cable adapter cards	10.000 DM
- 4 fibreoptic transceiver	11.000 DM
- 500 m fiberoptic cable	5.000 DM
	-----
total	39.000 DM

The prices compare well to a network with expensive remote repeaters.

## CONCLUSIONS

From a manager's viewpoint the MAN Technologie has introduced a proven local area network with an advanced technique. The solution is economic. The first goals could be reached.

The next steps towards integration will be the implementation of the TCP/IP protocol on all computers to enable file transfer between machines of different vendors.

The choice for this individual solution IEEE 802.3 is not a final one. It is obvious, that new developments respect existing standards. The greater the installed basis, the greater the pressure, to build gateways to other solutions from the beginning.


Hopefully this paper may encourage potential users.

## REFERENCES

- /1/ Raabe, Georg-Peter, (1986) Planet in der Universität Passau. State of the Art, 2/86 63-74
- /2/ Suppan-Borowka, Simon, (1986) MAP Datenkommunikation in der automatisierten Fertigung. Datacom Buchverlag, Pulheim
- /3/ Bux, W. (1984) Performance issues in local-area networks. IBM Systems Journal, vol.23, no.4, 351-374
- /4/ Holler, E. (1986) Ein LAN für ein Großforschungszentrum: von der Planung bis zur Realisierung. Datacom, Pulheim, 2/86, 4-21

## GLOSSAR

CADAM	System for computer aided design, Release 19.1, developed by Lockheed
CATIA	System for computer aided design, Release 20.2. developed by Dassault
CSMA/CD	Carrier Sense Multiple Access with Collision Detection; access method for IEEE 802.3 and Ethernet
Ethernet	Local area network, developed by Digital Equipment Corp., Intel and Xerox Corp.
EQINOX	EQINOX 7000: System for computer aided drafting (DRAFT) and interactive generation of NC-code (NCG), Applicon Schlumberger GmbH, Frankfurt
IEEE	Institute of Electrical and Electronic Engineers, New York
MAP	Manufacturing Automation Protocol: Local area network, introduced by General Motors Corp.
MARC	MARC: General purpose finite element program, Revision K2-2; Marc Analysis Research Corp., Pala Alto, California
NASTRAN	MSC-NASTRAN: Finite element program, Version 65; MacNeal-Schwendler-Corp., Los Angeles, California
PATRAN	PATRAN II: Finite element pre- and postprocessor, Version 2.0A; PDA Engineering, Santa Ana, California
TCP/IP	Higher level transport-protocol, used in many networks.

	HP 3000	DC02/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

**Networking: Will Today's Choices Be Needed Tomorrow?**

-----

Presented at the European Conference  
of Interex HP 3000  
Vienna, Austria  
23 March 87  
Scott Brear  
MICOM Systems, Inc.

I. Introduction

Over the past few years there has been a great deal of concern amongst data communication professionals regarding the long term viability of their networks. That is, will the investment being made today to answer today's needs be a true long term investment or will the equipment reach technological obsolescence far short of its operational life? To be more specific, most modems and multiplexors installed 10 years ago still FUNCTION as they were intended, but smaller size, greater diagnostic capabilities and more clever multiplexor features have oft times lured us into replacing the "old work horses" before their time. To complicate matters more, these product improvements are arriving on the market at an ACCELERATED RATE!

Perhaps your data communication needs are small. If so, do not think that you will escape the impact of these issues. After all, the usage of the word "network" transcends size, as NETWORK is usually defined. For example:


From DATAKONTEXT's Pocket Dictionary of Data Communications, they reference the DIN definition as "the assembly of equipment through which connections are established between terminal installations."

From MICOM Systems' Pocket Glossary, network is defined as "an interconnection of computer systems, terminals, or data communications facilities."

Large or small, networking is required because we are seeking CONNECTIVITY between various devices so they can EXCHANGE INFORMATION or, more commonly, SHARE RESOURCES. Both changes in telephone administration SERVICES and TARIFFS and IMPROVEMENTS IN TECHNOLOGY have inspired the data communications industry toward continuous product innovations which, in turn, have promised (and generally delivered) improvements in the cost, speed and flexibility of our resource sharing facilities.

With the emergence of personal computers and more powerful "super" mini computers, the term "resource sharing" no longer is solely associated with gaining access to and sharing a single central processor. Rather, we are now faced with a MULTITUDE OF NETWORKING CHALLENGES based on the exponential growth in the number of resources. Resources have been distributed to the smallest offices and to desktops worldwide creating all sorts of connectivity issues relating to intermachine COMPATIBILITY both on a LOCAL and WIDE AREA basis. Indeed, we would probably all agree that user requirements have been evolving from being "COMPUTER CENTRIC" to being "NETWORK CENTRIC."



	HP 3000	DC02/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

How will we address these issues? Will we install several diverse networks only to find that, while they do the intended function, they do not provide for future inter-networking capability? Will we install products that have the LOOK OF LAST YEAR'S MODELS too soon? This paper reviews these issues by way of a case study, and then suggests the application of methods and product concepts which will provide for FLEXIBLE NETWORK EVOLUTION.

## II. Alternatives Evolve

A dozen years ago, when I was faced with "networking" my first H-P 2100 computer, networking meant arranging access to the computer from several terminals within the building. The choices for our first (LOCAL AREA) network were easy: direct COPPER WIRE for nearby terminals, LIMITED DISTANCE LINE DRIVERS for medium distances within the building, and acoustic couplers or "real" MODEMS for those occasional users located away from the premises. At the same time, similar "networks" already existed separately servicing both IBM and DEC users. In those days three networks were not a problem, considering the realities; that is, a few "dumb" 300 BAUD terminals (1200 was fast then!) required NONSELECTIVE dedicated connections to a few CENTRALLY LOCATED computer resources.


### LOW COST MULTIPLEXING

Then, the obvious occurred. Additional users wanted to join their colleagues in the new world of remote CPU access. This raised a couple of new issues. First, a FAST GROWING NUMBER of individual LEASED or dial lines to distant divisions were now required to serve the growing user base. Well, this one was relatively easy to solve by installing low cost MULTIPLEXORS between the computer center and our remote divisions, creating our first WIDE AREA NETWORK (WAN). Not only did we REDUCE our leased line count to one per division, but we also gained the operational advantages of having ERROR CONTROL, DIAGNOSTICS and DATA MONITORING on the multiplexed link!

Furthermore, in those situations where EVEN ONE dedicated line was too expensive, but data traffic volume permitted, we installed several MULTI-POINT multiplexors. We thus retained the efficiency and error control benefits of the POINT-to-POINT units on LOWER TARIFFED multi-point lines. Of course, there were many others who also were discovering how easy it was to apply the newly available product technology to basic data communication problems. We all felt clever, indeed!

### PORT CONTENTION SOLUTIONS

The second issue revolved around the problem of have having TOO MANY USERS desiring connections to TOO FEW COMPUTER PORTS. The immediate solution was a crude "patch panel" whereby users would request computer center personnel to MANUALLY connect their direct or multiplexor-derived terminal connections when a computer port became available. Sound familiar? Some of us may fondly remember this as an early example of what we THEN THOUGHT was "complex" networking in the ASYNCHRONOUS terminal domain. At least we were able to interconnect the multiplexors with the patch panels in our growing network; it made sense as it was a simple solution to the problem at hand.

	HP 3000	DC023
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

We soon tired of patch panels as being too slow and costly. At least for the vast majority of users within the building complex, the solution was clear - upgrade to a PBX. No, not a PBX designed specifically for data, they came later. Rather, we were then converting the office facilities to a digital voice PBX (data capabilities were not then available) and the old "STEP by STEP" Strowger PBX was made available to us as surplus. Surplus indeed! We jumped at the chance to automate the patch panel function and it worked well. Users simply retained their older rotary dial telephone instruments and contended for "trunk lines" which were, in fact, ports on the minicomputers' front ends. Users were informed of unavailable ports via a "busy tone" rather than the embarrassed explanation of a computer center operator.


Not long after we put the Strowger DATA PBX into service, "real" purpose-built DATA PBX units came on to the market ... and we embraced them with glee! An important benefit to our operation, for example, was the INTEGRATION of our multiplexor units INTO the PBX equipment. We eventually had several bays of DATA PBX equipment serving the growing (but still computer-segregated) population of asynchronous terminal users. Connectivity to our various computing resources VASTLY IMPROVED as we further expanded our networks.

#### NEW SWITCHING SERVICES FILL A NEED

Again, we faced the future with heads high, having smoothly evolved our network to "state of the art." However, at this same time, we were also faced with an ever increasing number of geographically RANDOM users requesting access from customer sites and small sales offices around the world. They COULD, in most cases, use the PUBLIC SWITCHED NETWORK (PSN), but they needed more reliable AVAILABILITY and costs were excessive. Individual leased lines were also either too expensive, operationally impractical, or unavailable. Fortunately for our users, a new type of network service came into the public domain - X.25 PACKET SWITCHING NETWORKS. We tried it and we liked it!

This second WIDE AREA NETWORK, supplementing our existing, dedicated WAN multiplexor network, interfaced well with our LOCAL AREA NETWORK as they all used common interfaces. Indeed, our DATA PBX LAN systems even provided INTEGRAL ASYNCHRONOUS PADs for interconnection with the X.25 network! Of course, one of the best features of X.25 was that we needed no large capital outlay for backbone equipment - PADs were the only devices we needed to supply, and service costs were directly related to the usage of our random user base. Management was beginning to take favourable notice of our cost-effective efforts!

Incidentally, X.21 CIRCUIT SWITCHING services became available in certain European countries at this time. X.21 gave us a pay-as-you-go tariff similar to the X.25 concept, while providing a "real" circuit availability expected with full-time leased lines. Circuit switching has been strongly supported in some countries (notably Scandinavian) and, accordingly, the pricing in such countries is less than X.25. The reverse will be found in other cases. Pricing aside, for certain applications, X.21 was the ideal solution; and our STATISTICAL MULTIPLEXORS were configured accordingly.

	HP.3000	DC02/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## THE FIRST WORK GROUP REQUIREMENT

One day a new issue arose. A division which had required only communication with Head Office via our MUX WAN, now had INTERNAL data communication needs unique to that division. Its own offices and plants needed a communications system allowing for inter-office message and order processing with VERY FEW communication needs with Head Office. Whilst their existing links to the main network remained in place, we installed a separately-dedicated WAN system of SWITCHING MULTIPLEXORS which provided a cost efficient asynchronous network of terminal users. Again, we felt clever. After all, who would have then foreseen the NEED FOR THE SUB-NETWORK AND THE MAIN-NETWORK TO INTERCONNECT! The first rush of data communications was over....and we rested.

## VOICE/DATA INTEGRATION ATTEMPTS

Meanwhile back at the Head Office, our telephone administrators were beginning to consider the installation of special phones which would allow for SIMULTANEOUS TRANSMISSION OF VOICE AND DATA on the same wire pairs. This concept was of interest because wire pairs were getting more scarce, were inflexible and, frankly, voice/data phones seemed to be "the way to go." Implementors of these early attempts to merge voice and data soon learned that THERE WERE LIMITATIONS. For example, voice/data phones were generally based on the type of full-featured instrument offered only to the executives within a company and these same executives were usually not the ones who used asynchronous terminals. In such cases, companies were acquiring voice systems with a promise of long term future expandibility, but which really DID NOT FIT WITH THE COMPANY'S PERQUISITE CULTURE. None of us can afford this kind of mistake more than once!

Our company finally selected a different approach. We installed DATA OVER VOICE units throughout our buildings as they were far less costly, per terminal connected, than voice/data phones; we did not need to acquire a new or improved main voice pbx; and we still gained the two functional benefits of sharing wire pairs with voice and having user relocation flexibility - DOV could be installed WHEREVER WE HAD AN INEXPENSIVE TELEPHONE. Most importantly, we had implemented a system which was flexible enough to take us well into the future. In fact, the DOV equipment, integrated with our DATA PBX systems, significantly complemented and extended the operational life of our existing voice system. So far, we had stayed on the road to harmonious network growth whilst providing flexible, cost-effective connectivity between users and their resources. Remember....that was our objective!

During this same time, new voice technology was making other inroads into the end user domain. Several suppliers were offering VOICE MULTIPLEXOR products that promised to reduce the cost of leased lines by carrying more than one conversation on one physical telephone line. One promised integration within the data network and the other stood alone, but they are both worthy of review here as they help to further demonstrate network growing pains.

As to the first product, users were encouraged to install DIGITAL VOICE devices (known as VOCODERS) integrated into their standard high speed data systems, thus significantly reducing voice

telephone expenses. The idea sounded great and many tried with the hope that THIS was the beginning of the voice/data integration they had long been promised. Well, the voice quality was just not right; that is, users could usually understand WHAT was being said, but the vocoder technology would not allow for CONSISTENT and FAITHFUL

REPRODUCTION of the inflection, tone and emotional content of speech - qualities generally found with a good telephone connection. As a result, these earlier products were not commercially successful. Fortunately, NEW TECHNOLOGY AND DESIGNS are now offering SIGNIFICANT IMPROVEMENTS in the VOICE QUALITY and COST of such products, as we shall see.

The second voice scheme referenced above (called TASI - Timed Assignment Speech Interpolation - on the transoceanic cable routes) used STATISTICAL MULTIPLEXING to interleave analog portions (or packets) of speech from various different conversations onto fewer physical lines. Ratios of 2:1 were achieved and the voice quality was good, but companies needed a number of physical lines between two distant locations to justify the equipment cost. Again, investment in this technology offered no evolutionary merger with the data network; it was a DEAD-END IMPLEMENTATION.

#### CROSS DOMAIN ACCESS EMERGES


It was not long before our various computer users wanted to ACCESS EACH OTHER'S COMPUTER BRANDS as well as their own. This time WE WERE READY. Our earlier decision to acquire a DATA PBX PORT CONTENTION device now could be used to SWITCH TERMINALS amongst diverse CPU resources. By using appropriate tables, we could even establish the specific CPU(s) which the terminal users could access.

Access to IBM synchronous applications was also possible via the DATA PBX. we allowed our asynchronous terminal users to connect to an INTEGRAL PROTOCOL CONVERTER feature which, in turn, was viewed as a synchronous control unit to the IBM processor. Indeed, even our mini's, when emulating asynchronous terminals, could access the IBM via the protocol converter function.

Finally, we reached the point where many of our divisions had their own DATA PBX systems and they wanted to have regular access with each other AND Head Office. An INTERCONNECT feature was available which provided for configuring the various DATA PBX systems such that their attached terminals could, under MANAGEMENT CONTROL SOFTWARE, access terminals and ports at other locations using a SINGLE address.

We were continuing to EXPAND our networks and, up until now, they still seemed to be well integrated, providing a sound foundation for the future. But, we were beginning to miss something. Some of our network components had good MANAGEMENT CONTROL, while others had little or none; and we concluded that, before too much more growth occurred, we would need comprehensive vendor-supplied "tools" to ORGANIZE AND MANAGE the network.



	HP 3000	DC02/6
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## THE PERSONAL COMPUTER ARRIVES

You are all familiar with the story. Personal computing SIGNIFICANTLY IMPACTED the provision of data processing and data communication services with respect to both the large "mainframe" people and the "mini" people within our company. At first, of course, it was not a problem as PC users acquired their machines and quietly applied them to a SPECIFIC DESKTOP APPLICATION. But soon these "quiet users" began to want access to the mini and mainframe databases (and each other) and the connectivity issues grew from there!

At first, solving this problem was not too difficult as we only had to cause the SMART PC to emulate a DUMB TERMINAL (a world with which we were very familiar) and then insert the PC into the DUMB TERMINAL NETWORK which we had so painstakingly built over the years. Right? Not quite! Whilst the earlier PC users were happy with this scheme, THEIR growing sophistication and the increasing complexity of their SOFTWARE applications soon outgrew our network's abilities. Our trusty DATA PBX could not transfer data at speeds greater than 19.2K Bits - a loafing speed when compared to rates in excess of 100K Bits possible with PC's!


To compound the "problem," PCs were becoming more powerful - seemingly so every month! Now we had PCs with MORE RAM MEMORY and FAR MORE POWER (in terms of MIPS, or millions of instructions per second) than many super-minis or large mainframes we installed less than 10 years ago. Clearly, a unique network solution was needed.

## LOCAL AREA NETWORKS - PC STYLE

One solution used by many PC users was (and still is) what some may call "SNEAKERNET" or the running about an office complex swapping disk files between PC's. There were obvious short falls to this "solution" such as ....how does one use SNEAKERNET to swap hard disk files? Worst of all, SNEAKERNET provided no integration path into our existing data network. To be sure, we quickly discouraged THIS network scheme.

Computer and Local Area Network companies soon supplied a better answer in the form of ETHERNET and other cable-based high speed local area networks DESIGNED SPECIFICALLY for the attributes and needs of the PC world. These LANs were purpose-built for a new breed of CPU user and they generally delivered as promised - PCs were able to communicate with one another at "PC speeds" allowing for resource sharing to the same extent that we have been accustomed to resource sharing as minicomputer users. In fact, we were concerned. Did these new LANs portend the DEMISE OF OUR DATA PBX network? Was our large installed base about to be rendered obsolete?

But what about our grand corporate network plan whereby ALL users can INTERCONNECT with each other and each other's applications? Local area networks began to show a flaw; that is, there were very few ways to efficiently provide cross-domain access between one work group and others - especially if the groups were separated by large distances. This has, in turn, spawned another innovative network technique that users in Europe may call "DHL-NET," or the

	HP 3000	DC027
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

sending of disk files via the overnight express postal service. No, cable-based LANs were not going to be able to fulfill ALL of our networking requirements.

#### SYNCHRONOUS REQUIREMENTS KNOCK

Whilst we were busy building a superb asynchronous network to serve our mini users, the IBM mainframe group began to request similar assistance. The logical move, one would think, would be to blend the two worlds - async and sync - into one single homogeneous network. This was practical from a network "backbone" perspective as the leased telephone line facilities were, by far, our GREATEST SINGLE NETWORK EXPENSE. However, the nature of the respective data AND user needs PRECLUDED SYNCHRONOUS use of our DATA PBX SWITCHING network which had been serving our asynchronous users so well. Certainly, portions of our various WAN networks WERE USED by the IBM group to gain economies of scale in using the leased facilities. For example, statistical multiplexors and X.25 PADS could accommodate synchronous data, but we could usually go no farther toward integration of the sync and async worlds.

#### III. LESSONS LEARNED...TO BE LEARNED

We have just learned how a typical company evolved its data communications system over a dozen years based on the various requirements of its Head Office, remote divisions and customers. Did it sound familiar? It should have, as our experiences were similar to many in our industry. We were rather fortunate to have made decisions that retained a measure of harmony and yet provided a large degree of connectivity amongst the various components ... ALMOST!


Now, let's take a moment to review a few KEY ISSUES which may impact YOUR NETWORK'S FUTURE:

#### COMPLETE CONNECTIVITY SLIPPING AWAY

As long as all the terminals and computing resources had asynchronous V.24 (RS-232C) interfaces, we were quite able to connect virtually any device with any other device. CONNECTIVITY WAS HIGH! However, when transmission speed requirements began to creep over 19.2 kb and direct CPU bus interfaces became more prevalent, the "rules," as we knew them, changed dramatically and we were seeing the notion of "GLOBAL CONNECTIVITY" slipping away. We began to search for products which would allow us to maintain our "golden fleece" network.

#### MANAGING THE NETWORK

Global connectivity is an attainable goal, but how do we propose to manage the resulting network? Such an effort will go well beyond the abilities of a small staff to real-time manage the modems, switches, multiplexors, PADS ....not to mention MANAGING THE MEDIA provided by the various telecommunication authorities worldwide....without the assistance of a dedicated COMPUTER-BASED RESOURCE MANAGER. The software effort to perform this function will need to be elegant, indeed!

	HP 3000	<b>DC02/8</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

#### FEAR, UNCERTAINTY AND DOUBT (FUD)

Many say that a complete "DO EVERYTHING" network solution will never arrive. To be sure, vendors have been and will continue to tout the "ultimate solution," but you should consider the impact false starts have on your network plans and programs. Do not lose sight of OTHER TECHNOLOGICAL DEVELOPMENTS which may adversely influence the performance of any such "ultimate solution" you select. For example, consider how the new ISDN networks may impact your choice? Will ISDN render your choice obsolete, even in part? Or, What impact will FIBER media have on either your WAN or LAN network plans?

Consider the new 32-bit PCs such as those based on the powerful Intel 80386 processor chip. Will the processing speeds of these units demand FAR HIGHER TRANSFER RATES than you are able to handle through your new "solution?" Finally, what will be the impact of newer voice technology? The trend is clear - VOICE will become more and more a part of our data networks when high-quality low bit rate technology arrives. Will your system have room for corporate voice traffic?

#### IV. BACK TO BASICS

Perhaps we should pause to remind ourselves of some BASIC CRITERIA which should influence any networking decision. They should be weighted as to how important they are to YOUR organization. Step softly, DO NOT FOLLOW the TRENDS unless such trends DIRECTLY APPLY to your needs!

- o Consider the ECONOMIC and POLITICAL situation within your environment and how this may impact TARIFF rates from telephone authorities. We have seen rates double - thus quickly "unjustifying" a major network component.
- o Consider PERFORMANCE. Is response-time a valuable feature of your network which you "sell" to users? How much are you willing to pay for good response in terms of hard currency and, perhaps, less reliability?
- o Remember CONNECTIVITY? Inter-operability of your network components and users may be essential. For example, the widespread use of a corporate ELECTRONIC MAIL system requires, obviously, very high inter-connectivity...what different LAN and WAN protocols are in use and how do you provide connectivity between them?
- o Who needs to access all of the RESOURCES? A comprehensive security plan may be needed along with a network manager which can insure that only the "right" people can access the appropriate data base.
- o What is happening to TECHNOLOGY? Are you considering advances learned from your peers, the press and trade shows? Should your equipment be flexible enough to change and grow as STANDARDS and network technologies advance?

- o RELIABILITY and AVAILABILITY should be cornerstones of any network, but how much weight should they have with you? For example, AVAILABILITY may be key to a public data base service whereby users may randomly chose to search for reference data during the day. On the other hand, RELIABILITY may impact how much redundancy your network requires in order to provide the requisite availability.
- o Finally, do not lose sight of the need for COMPATIBILITY. As you make equipment selections, decide whether such equipment must be compatible with your existing network components. In several cases, manufacturers have provided updated products which have RETAINED their ability to be BACKWARD COMPATIBLE with earlier generation products.

#### V. WELL THEN, WHAT SHOULD WE DO NEXT?

Armed with a renewed awareness of KEY ISSUES which may impact your network planning and with full consideration for the BACK-TO-BASICS CRITERIA, you should begin planning the "NEW NETWORK." Bare in mind that "new" does not necessarily mean replacing all your existing equipment. Rather, it means organizing and managing future evolution and growth with a NEW FRESH PERSPECTIVE as to notions of comprehensive inter-network CONNECTIVITY.

But first, you must conduct a BASELINE EVALUATION of your network. In simple terms, this means a review of all components of the network and sub-networks FROM THE GROUND UP. What were the original requirements; were they satisfied and with what equipment and services; what fell short and why? Do not bypass a component or method because "that is the way it has always been done." Assume nothing. Only when this exercise is finished will you fully appreciate the state of your network(s) and the inevitable PATCHWORK that has, no doubt, developed over time.

Furthermore, it is essential to conduct an INVENTORY of the PROCESSING RESOURCES ACCESSED along with an inventory of the equipment and services which PROVIDE ACCESS for the user groups. Again, you need the WHOLE "picture."


#### TYPES OF NETWORKS

When conducting your inventory, GROUP YOUR NETWORKS by type. For example, how many and what types of cable-based systems are installed? Where are they installed? What inter-connectivity exists between them - do not assume it is NOT necessary. If you have DATA PBX networks, on the other hand, can THEY interconnect with one another? How many different PBX systems are installed? POINT: when you have concluded this review you may be surprised to learn how many UNNECESSARILY DIFFERENT TOPOLOGIES are deployed within your organization supporting only a few FUNCTIONAL TYPES of networks.

#### TEST NETWORK APPLICABILITY

Consider whether you have applied the appropriate topology within your networks. Perhaps a DATA PBX system is being used, in part, for PC-to-PC file transfer - a task well served by the PBX. However, when the PCs require a centralized FILE SERVER function,



	HP 3000	DC02/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

it would be unwise to continue with the PBX as the SOLE NETWORKING VEHICLE. Rather, the file server requirement would be better served by a much faster (and more expensive per port) cable-based LAN. Moreover, the PBX should be retained AS PART OF THE TOTAL SOLUTION should the PC's require access to others NOT WITHIN LAN CABLING DISTANCE of each other. Note: this latter point brings up a CONNECTIVITY ISSUE. Search for ways to apply your NEW network perspective; e.g., use LAN's for the locally-situated PC's WHERE SPEED IS CRITICAL and consider BRIDGING regional LAN's via a DATA PBX GATEWAY. Who knows, maybe you too can eliminate "SNEAKERNET."

Do not forget the VOICE NETWORK. Dedicated, expensive "hot" lines may be needed for critical voice traffic between plants, but frequency of the critical communications may be only twice per day. If DATA TRAFFIC exists between the two plants, why not consider using products which can allow voice conversations to be carried on the data network? Conversely, where excess WIDEBAND VOICE capacity is available, why not install data MULTIPLEXORS?

Let's look again at our first network example, the one using MULTIPLEXORS. Are point-to-point systems in place when traffic considerations only require less expensive MULTI-POINT? Do you have the equipment which will allow for conversion from one topology to another?

#### HARMONIZE SIMILARITIES AND EXPLOIT DIFFERENCES

Make a concerted effort to BLEND SIMILAR REQUIREMENTS and networks together. Such a move may seem obvious, but one would be surprised to learn how many applications are wastefully running on separate networks, because of erroneous initial ASSUMPTIONS concluding that - but for a simple change - they are not able to run together. If the INFLEXIBILITY of NETWORK EQUIPMENT is blocking your way, find equipment which CAN run various applications. Frequently, providers of such flexible equipment for TODAY'S NEEDS are also clever enough to have foreseen the requirements you will have TOMORROW!


Differences will exist. As was mentioned earlier, there will always be that "unique application" requiring special network hardware or software which currently is not used in your operation. FIGHT THE URGE to FORCE a fit. After all, you may find that such a unique application will INSPIRE THE DEVELOPMENT of an entirely new networking concept; for example, remember how the need to share a limited quantity of CPU ports inspired the development of the DATA PBX!

#### VI. THE FUTURE LOOKS BRIGHT!

We have discussed a number of growth issues, relating to network CONNECTIVITY, in the context of a typical company's experiences; and we have reviewed general recommendations as to how to work through these issues. Now, let's have a look at some SPECIFIC SOLUTIONS which are available.....perhaps we should also have a peep into the future.

#### GATEWAYS OFFER CONNECTIVITY AND GROWTH PATHS

The need for true INTER-OPERABLE GATEWAYS has been shown throughout this presentation. Gateways must not only provide the minimal

	HP 3000	DC02/11
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

PHYSICAL ability to cross network boundaries, but a true gateway must allow the attached devices to operate in their NATIVE MODE constrained only by the data transmission speeds permitted on the associated media. For example, although VOCODERS allowed voice traffic to operate over data lines, they failed as true gateways as the speech quality could never have been considered NATIVE MODE.

Speaking of voice products, a gateway of particular interest is the DMI/3000 product from Hewlett-Packard. DMI, or DIGITAL MULTIPLEXED INTERFACE, was conceived by the American Telephone and Telegraph Company (AT&T) as a non-proprietary interface standard for transporting data communications over private digital telephone networks. The DMI/3000, consisting of a two-card set and associated software for installation into the backplane of an HP 3000 computer, provides for the H-P system connection of up to 23 terminal devices (using the American T-1 standard, at first introduction) DIRECTLY from an ISDN link. The DMI/3000 promises to solve many of the connectivity problems related to the multitude of proprietary interfaces which abound with DIGITAL VOICE PBX systems and their "data modules."

Gateway features can reside in software as well. Until the LAN transport standard, TCP/IP, was accepted by equipment vendors such as Hewlett-Packard and MICOM Systems, for example, there was very LITTLE practical COMMONALITY amongst cable-based LAN systems. As a result users were severely limited in connectivity capabilities; i.e., they left the high speed IEEE 802.3 environment, transversed a slower V.24 "gateway" or converter and went back into another 802.3 LAN. A newer and faster standard than TCP/IP, ISO TP4, promises to improve such connectivity even further.

In another development, MICOM has recently announced a product which provides a BRIDGE between its IEEE 802.3 cable-based LAN and its DATA PBX products. The NTS470, coupled via twisted-pair wiring (up to 1.5KM) to a companion module in the DATA PBX, allows up to 8 terminal users accessing the DATA PBX to directly select "ports" on the associated LAN and, of course, LAN Terminal Server users can access processor and WAN ports on connected to the Data PBX. Since the NTS470 supports the TCP/IP standard, it offers high "interoperability" (remember that word?), to users of BOTH the cable-based LAN and the DATA PBX! Finally, as connectivity requirements grow, additional NTS470 gateways can be added between these two "sub" LANs of your network.

#### MODULAR PRODUCTS

One of the few consistent characteristics of our networking future is that THERE WILL ALWAYS BE CHANGE! With this in mind, acquire networking products that, where practical, are flexible enough to change when STANDARDS, your NETWORK REQUIREMENTS or TARIFFS CHANGE. Installing network "BOX's" around the globe is difficult enough - why not find a way to MINIMIZE THE EFFORT?

For example, you may wish to install a pair of multiplexors on a LIGHTLY USED data link only to decide several months later that these same multiplexor locations should be JOINED WITH OTHERS in a larger network. Or, perhaps you want to connect these locations to an X.25 PDN. For these reasons it would be wise to select the equipment supplier who could offer you a BASIC COMMUNICATION

PRODUCT which could "change personalities" simply by the exchange of FIRMWARE CARTRIDGES. This product concept would then allow you to install the above-mentioned units WITH MULTIPLEXOR FIRMWARE, knowing that, should requirements change, you would merely need to send out new cartridges - NOT replace the entire unit!


Correspondingly, HARDWARE MODULARITY is equally important. Equipment has been available which provides for EASY CHANNEL EXPANSION by simply plugging cards into a basic chassis. This has applied to products as diverse as DATA PBX's and MULTIPLEXORS. However, new products are becoming available which are stretching the hardware modularity concept EVEN FURTHER.

MICOM, for example, has recently announced a multiplexor which combines an ASYNCHRONOUS MULTIPLEXOR function with a SYNCHRONOUS WIDEBAND TIME DIVISION MULTIPLEXOR. Consider the benefits of being able to add one of these functions to the other whilst in the field. Or, in the case of another modular product, consider the benefits of mixing asynchronous and synchronous PAD's in the SAME CHASSIS unit or mixing either one with a packet switch function!

Remember when we discussed attempts to INTEGRATE VOICE requirements with data networks? Latest developments in LOW BIT RATE (sub 32kb) voice technology allow speech patterns to be FAITHFULLY REPRODUCED over digital channel speeds as low as 9.6kb. While voice products have been available which provide reliable speech channels at 32kb or greater, they were usable on only the highest speed channels - sometimes with LITTLE ROOM left over for data. The NEW TECHNOLOGY, on the other hand, offers plug-in MODULAR digital voice capabilities to the SMALLER networks. Now, VIABLE voice/data integration is possible on network facilities of LESS THAN 1.5Mbits/sec!


Let's peep into the FUTURE a bit. Suppose we develop this concept of MODULARITY into a networking environment. What would the ideal product look like? Imagine, if you will, a product which could START as a medium speed point-to-point multiplexor and GROW into a WIDEBAND, MULTI-NODE networking vehicle. Notice that I am suggesting the product should "GROW into" - not be replaced by - a "networking vehicle." Such a product should offer MORE THAN STATISTICAL MULTIPLEXING. Indeed, it should have the MODULAR capability to offer a wide assortment of NETWORK GATEWAYS and interfaces such as:

- o X.25 PAD's and Switching, not only for end-users, but for the backbone BETWEEN the vehicles.
- o Digital Voice, at compression rates tailored for specific voice network requirements.
- o IBM Protocol Conversion, for BSC and SNA applications.
- o IEEE 802.3 LAN, with TCP/IP and other standards as required.
- o Multiple High Speed Interfaces, with both the CEPT 2 Mbits and American 1.5 Mbits speeds.

	HP 3000	DC02/13
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

To tie all of this together, one would require EXTENSIVE NETWORK MANAGEMENT. Network management does not mean having one clever PC program to run the X.25 network and another to run the statistical multiplexor network. Rather, this product concept, as it grows to its full capabilities, should have a comprehensive SINGLE interface available to the network manager allowing access to and control of ALL of the DIFFERENT ENVIRONMENTS.

Such a product, if available, would give you a GROWTH and FLEXIBILITY capability unheard of in today's datacom environment! More importantly, it should be backwards COMPATIBLE with most products available today so EQUIPMENT ACQUIRED TODAY would be just as usable TOMORROW. Indeed, with products such as this, not only will ".....today's choices be needed tomorrow..." but, TODAY'S choices will be the CORE OF TOMORROW'S PRODUCTS!



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

--	--



**THE VICORP VIEW OF VIDEOTEX:  
A UNIFORM INTERFACE TO DISTRIBUTED SERVICES**

Stuart R. Patterson  
Marketing Manager  
VICORP Marketing GmbH

**Introduction**

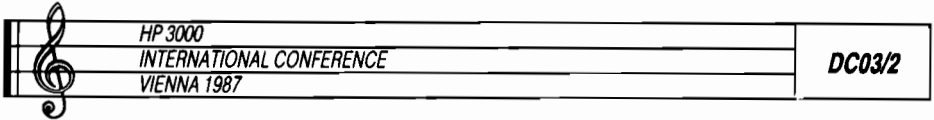
Thanks to the media, many people now think of videotex as a French phenomenon. If we analyze, however, the true role of "videotex" within the data processing (DP) environment, it is clear that over time videotex-type solutions will appear around the world (albeit perhaps under another name).

This paper presents videotex as a Uniform Interface to Distributed Services (UIDS). This view is based on approximately 100 installations in 15 countries of a videotex-oriented communication system known as "BETEX". Following a brief review of the French success, and the misleading conclusions being drawn from it, the author analyses the four interfaces which a complete videotex system must manage:

- \* the application to network connection;
- \* the host computer to network computer interface;
- \* the network computer (or node) to terminal interface;
- \* the terminal to user interface.

**1. Learning from France**

Few people remember that the French concept of "telematique" was introduced in 1978 well before Francois Mitterand and the Socialist ministers assumed office. It is useful to review, in fact, the seminal work by Nora and Minc, The Computerization of



Society, (1) which was the foundation for the Teletel program and the fame of the Minitel. Their proposal for subsidised access to a panoply of online services grew out of one, relatively simple argument: that the average Frenchman (and perhaps French industry) would be passed over by the wonders of information technology if the state did not intervene to make online services available in a standardized yet attractive form.

Thus, the goal of the Teletel program was to accelerate the penetration of DP services in society. Many analysts, however, confuse the means with the ends by focusing on the tools -- Transpac, the Annuaire Electronique and the Minitel -- chosen to reach this end rather than the end itself. The marketing mix which the French PIT has so skillfully sold to the public should not blind us to what they have accomplished in DP terms: i.e., to make one terminal (with one syntax, language, and presentation standard) the interface to a network of geographically distributed computers which offer a variety of applications (using the same dialogue) to the user. Would that all corporate DP departments could boast such an achievement!

## 2. Videotex as Distributed Data Processing

If the Nora/Minc view is valid, then videotex has a distinct place in the world of distributed data processing (DDP). Videotex services are, in fact, a distinct subset of DP services, not an alternative. The DP services currently made available via public videotex networks may be broken down into three categories (most private systems also include all three):

- 1 - information delivery
- 2 - interactive communication
- 3 - online transaction processing.

Each of these groups of services is external in orientation and depends on distribution to be effective. Services which fit under these headings, therefore, must be distinguished from the original driving force behind DP -- the automation of internal, administrative tasks such as accounting, correspondence, and personnel policies. When we compare these two groups, in fact, we find important differences (see Table 1).


As Table 1 suggests, the "administrative" services merit -- indeed often require -- unique terminals, languages, networks, applications, and even CPUs in order to function efficiently. "Videotex-type" services, on the other hand, generally use low-cost terminals, the language of everyman, public networks, consistent applications, and standard CPUs.

**Videotex can not serve, and should not be seen, as an alternative to the intensive and often user-specific systems used for internal, administrative tasks. That being recognized, we must ask ourselves whether historical DP solutions are appropriate for the provision of services to a large number of distributed users.**

### 3. The Interface Problem ... and Opportunity

Carrying the comparison presented above one step further, we might wonder how to develop a package or system which is optimized according to the characteristics of videotex or



	HP 3000	DC03/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

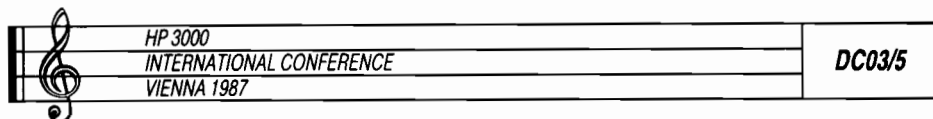
distributed applications. This approach reveals four interfaces, each requiring different adaptations (often country or hardware specific) and together demanding exceptional flexibility.

#### A. The User-Terminal Interface

The first and most easily understood interface is between the user and his terminal (see Figure 1). This is often referred to as the "human-machine interface." While many system elements contribute to what appears on a terminal screen, let us start simply with the neophyte's point of view. The concerns of the first-time user are clear and have been well analysed. Table 2 lists the questions the new user might ask alongside of the translations we in the industry might make for our product design team.

The table illustrates that this interface alone can be extremely complex from the service provider's standpoint. When we consider the current set of answers to such questions, the task becomes more not less difficult. Many of the concerns, for example, are addressed (or not, as the case may be) by the choice of presentation level standard. "Standard," however, is a misnomer as five are already widely used for videotex: ASCII, Prestel, Teletel, CEPT, and NAPLPS. In many videotex-type applications, the IBM 3270 might also be considered a "standard."

As a result of these options, the service provider is generally forced to decide, by the PIT or the system supplier, in which format his application will appear. But what if he wants the




same application to appear in several formats ... one for users with simple monochrome terminals and another for users with color graphics workstations? And what about tomorrow's "standard?" These valid concerns of the user and the service provider highlight the need for a very flexible or modular solution for the delivery of services.

#### B. The Terminal to Network Interface

The terminal for distributed services access usually has a modem in it or behind it but the simplicity stops there. If the service in question is truly "distributed," then it will be based on a network of nodes or access points. Thus, the connection of the terminal (where sits our neophyte user) is established with or via a network node (see Figure 2).

In the European videotex environment, these nodes are known as Videotex Access Points of URPs (in France, "Points d'accès videotex" or PAUs). For a distributed service of any name, these may be the nodes of a corporate datacommunications network. [2]

If the communication between the user's terminal and the network node is to be successful, then the two machines must speak the same language. Thus, one of the most important questions in the videotex world is: Which terminal protocol will be used? The Minitel user, for example, may connect to any PAU in France. On the other hand, he may not connect to any of the URPs used in Belgium or Denmark, the Prestel system in the UK, or the BTX nodes in Germany.

	HP 3000	<b>DC03/6</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

This machine-to-machine interface, then, is as important as the human-machine interface examined earlier, and the two are always related. Table 3 presents a list of the most common variables which can make or break a terminal-node connection.

Table 3 reveals the potential for difficulties in connecting a terminal to a network node. From an American point of view, the ability of the European PII's to effectively impose a solution to these difficulties is enviable. For a given corporation, however, the consistency of the public networks does not guarantee the means to provide this interface. Consider, for example, the case of a company having an installed base of IBM 327x terminals. If a service is designed for distribution via the videotex network, then a separate access path must be established for the successful connection of the 327x users.

With an installed base of millions of terminals of varying shapes and sizes, corporations must provide for the old as well as the new. As videotex (or distributed) services become more sophisticated, the delivery systems or servers (as they are known in France) must offer a simple version to the simpler terminals, without slowing the evolution of the service itself.

Once again, we see that the best solution to the interface problem is flexibility. The service provider who attempts to resolve the interface questions at the start is apt to encounter problems in the future. Instead, the goal should be to secure

the means to develop new solutions effeciently, if and when needed.

### C. The Network Mode to Network Server Interface

Datacommunications networks are growing in number and in sophistication at an astounding rate. Despite this revolution, the More and Minc analysis of 1978 may still be valid:


... It is based on the "law of the jungle," the strongest in this case being IBM, which has the best chance of providing these connections for most of the networks. [3]

Indeed, SNA is by any measure the predominant network protocol. And yet, the videotex networks are -- in a technological sense and particularly in France -- providing an enhancement or extension to the services available via SNA.

To help our neophyte, though, we need not control the overall network. We must instead make sure that a particular service is accessible via a particular node (see Figure 3). This ability is ensured as long as we restrict the user to one of two environments:

- 1 - that of one hardware manufacturer (e.g., SNA-compatible services are accessible via SNA network connections); or
- 2 - that of one videotex system (e.g., services implementing Prestel Gateway 2.3 may all be accessed via the Australian videotex network).

Unfortunately, limiting a service to a national or a proprietary network is contrary to the idea of a universally accessible


	HP 3000	<i>DC03/8</i>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

service. French marketing managers, for example, who want to reach the 2.1 million Minitel users now ask: "Can I make our SNA-based service for information retrieval (or other application) available via the Teletel network without recreating the application?"

The machine-to-machine interface which assures such connections is often referred to as a "Gateway." The gateway concept may also be described as an implementation of the layered Open System Interconnection model (OSI). In the videotex world, the German Bildschirmtext system is notable as an implementation of layers 4 (Transport) and 6 (Presentation) of the OSI model. Layers 1 to 3, of course, are implemented on the X.25 networks that form the backbone for the vast majority of existing videotex networks.

SNA is also a layered network protocol. SNA nodes, however, use a different protocol than the videotex network nodes in Germany (there, the protocol is called "EHKP"). Similarly, the Prestel Gateway protocols, of which there are several national versions, provide for the upper layers of the OSI model -- but they are not layered!

Here again, the interface is variable and becoming more so. Videotex-type systems are solving these dilemmas every day; some, however, are extremely limited in their ability to incorporate new or more extensive functionality as networks evolve.

	HP 3000	<b>DC03/9</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

#### D. The Network to Application Interface

Up to this point, we have examined only those interfaces which are confronted as a signal of bits and bytes is sent from a host, over a network, to a terminal. One last connection, perhaps the most important, remains to be made before our user can access the data or application he seeks: the connection between the application program and the network (see Figure 4).

Many of the questions posed by our eager neophyte (Table 1) must be addressed, in fact, at the application level. The network architectures, such as OSI and SNA, also impose requirements on the application in their upper layers (i.e., layers 5 to 7). Consequently, **the application developer who hopes to provide distributed access to his program must, in the absence of a service delivery package or shield, plan for various networks and terminal types.**

Some of the questions the developer must consider may be addressed elsewhere on the path to the terminal. In the past, however, the opposite has been true and applications developed for one network or terminal protocol or presentation standard are -- for all practical purposes -- restricted to that network, terminal and standard. Table 4 presents some of the objectives of the programmer which a system optimized for distributed services delivery might address.

Table 4 demonstrates the variety of choices a developer must make without some sort of communications front-end. **Many programmers**

have been forced -- in accordance with the hardware, operating system, or distribution network chosen by the service providers -- to fix these variables at the outset. In France, the most mature videotex market, this phenomenon has led to the appearance of RFPs for "2nd Generation" videotex systems. As one would expect, the typical requirements outlined in these RFPs are that the terminal type be variable or the utilities for frame creation and updating be provided with the system or other demands for increased flexibility.

The interface at the developer's level is where many of the decisions are made as to how to address the three other interfaces examined in this paper. System integrators often answer the multitude of interface questions for each system they "integrate." Perhaps they should instead turn their attention to developing high level tools enabling their clients to address such fundamental questions if and when they arise.

#### 1. The Complete Picture

When we combine the four interfaces described in this paper, we see the challenge of providing a uniform interface to distributed services (see Figure 5). Despite the seeming complexity, the figure is actually a simplified representation of the DDP landscape.

It is no surprise, therefore, that the 100-odd installations of BETEX often implement similar functions but are very rarely identical configurations of the system modules. All of them

simplify application development for the provision of a uniform interface by providing powerful, high-level tools enabling programmers to address the issues discussed in this paper.


## 5. Conclusion

In the middle of Nora and Minc's prescient analysis of the DDP revolution, we find the following:

The objective is to ensure open exchanges by allowing users to converse among themselves independently of their equipment. Otherwise, they could not use the hardware or services of another manufacturer. [4]

Videotex is contributing to the realization of this goal. Minitel users in France are spending significant sums of money "chatting" on the Teletel network -- none of them know or need to know which manufacturers or protocols are making the connection possible. At the same time, more and more corporate DP departments are using videotex-based solutions to provide a uniform interface to their distributed services.



	HP 3000	DC03/12
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

---

**NOTES**

- [1] The Computerization of Society, A Report to the President of France, Simon Nora and Alain Minc, 1980, The MIT Press  
(originally published as L'Informatique de la société, 1978).
- [2] Where a series of nodes does not exist and all distribution is being handled by one system, then the functions described here for a network node are applicable to the host system.
- [3] Ibid., page 27.
- [4] Ibid., page 74.

**TABLE 1**  
**Comparison Between "External" and "Internal" DP Services**

Characteristic	External/Distributed Applications	Internal/Administrative Applications
Number of users	Large	Small
Time of use	Around-the-clock	Work hours
Duration per logon	Short (<1 hr)	Long (several hours)
User profile	Untrained	Trained
Geographical	Highly distributed	Company/site specific
Data base use	20% of info. serves 80% of demand	Highly variable
Graphics	Often	Seldom
Transaction rate	Low	High

**TABLE 2**  
**New User Questions and Industry Interpretations**

New User's Question	Industry Interpretation
Does it look nice?	Presentation standard?
Does it speak my language?	Language for screens and prompts?
Is it easy to learn to use?	Menus, keywords, interactive messages, help screens?
Will it react the same way tomorrow?	Application syntax?
Am I restricted to text only?	Graphics (images/icons)?
Can I do more next time?	Intelligent terminals, telesoftware etc.?
Can I use it to type letters?	Integrate with other DA services?
How much do I have to pay?	How much do we have to pay?

**TABLE 3**  
**Significant Variables in the Terminal - Network Interface**

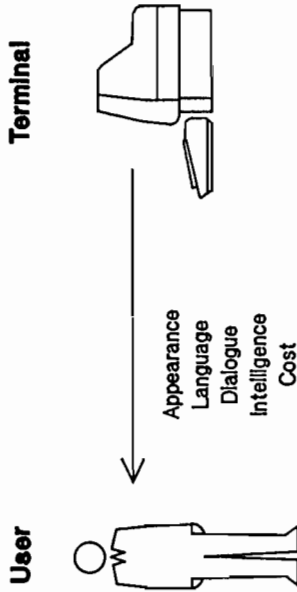
Terminal-Network Variables	Current Solutions
Medium or network type	Hardwire (direct) Telephone lines (dial-up) Cable X.25 (packet-switched) Videotex (access points) Value-added networks (VANs) ISDN
Transmission method	Asynchronous Synchronous
Transmission channels	Simplex Half-duplex Full-duplex
Transmission speed	1200/75 bps 300/300 bps 1200/1200 bps 2400/2400 bps 9600/9600 bps etc.
Terminal standard	ASCII (ANSI standard) Prestel Teletel CEPT NAPLPS

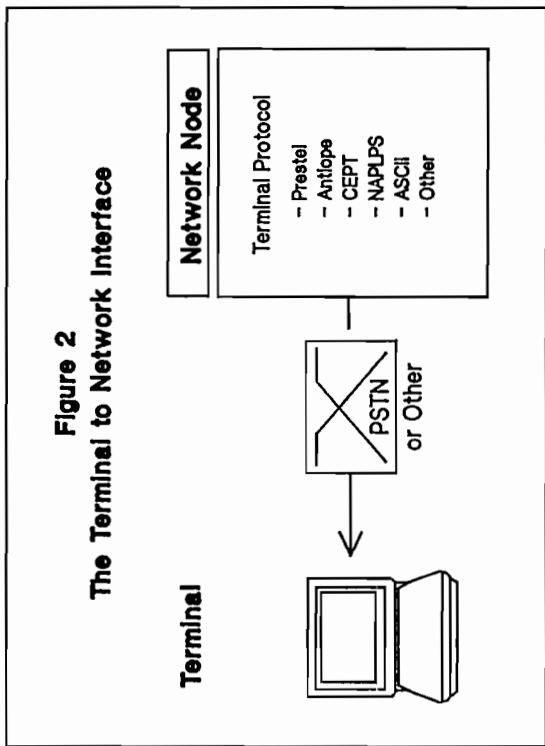
**TABLE 4**  
**Options Facing Developers of External, Distributed Applications**

Developer's Objective	Typical Options or Concerns
CPU or operating system flexibility	Front-end configuration; porting; etc.
Access path flexibility	Front-end communications processors
Familiar development tools	Command interpreters
Security	User ID codes; closed user groups (CUGs); passwords per page; independent databases; terminal or modem IDs; smart cards; etc.
Presentation standards; graphics	ASCII; Prestel; Teletel; NAPLPS
Navigation syntaxes	Prestel (*#, etc.); function keys (e.g., Minitel); 1-2 digit menu choices; keywords; interactive user prompts; etc.
User language	English, French, ...
User administration	Logon/logoff; usage records; real-time communication to the user; user-specific frames or data; etc.
Terminal type	ASCII (IBM, DEC, HP, DG, etc.); videotex (Prestel, Minitel etc.); block-mode (327x etc.); ATM; etc.
Frame/page creation (esp. for videotex)	Field definition; frame database; graphics composition; frame linkage; information updating; etc.



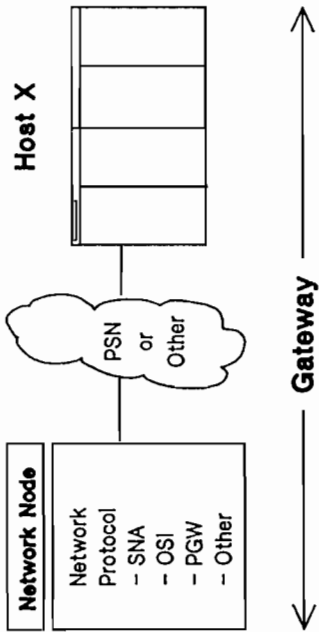
**Figure 1**  
**The User to Terminal Interface**







**Figure 3**  
**The Network Node to Host Interface**



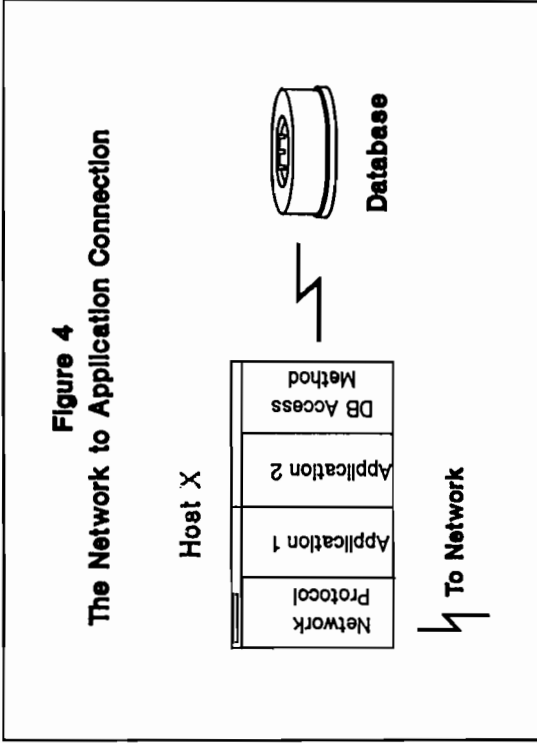
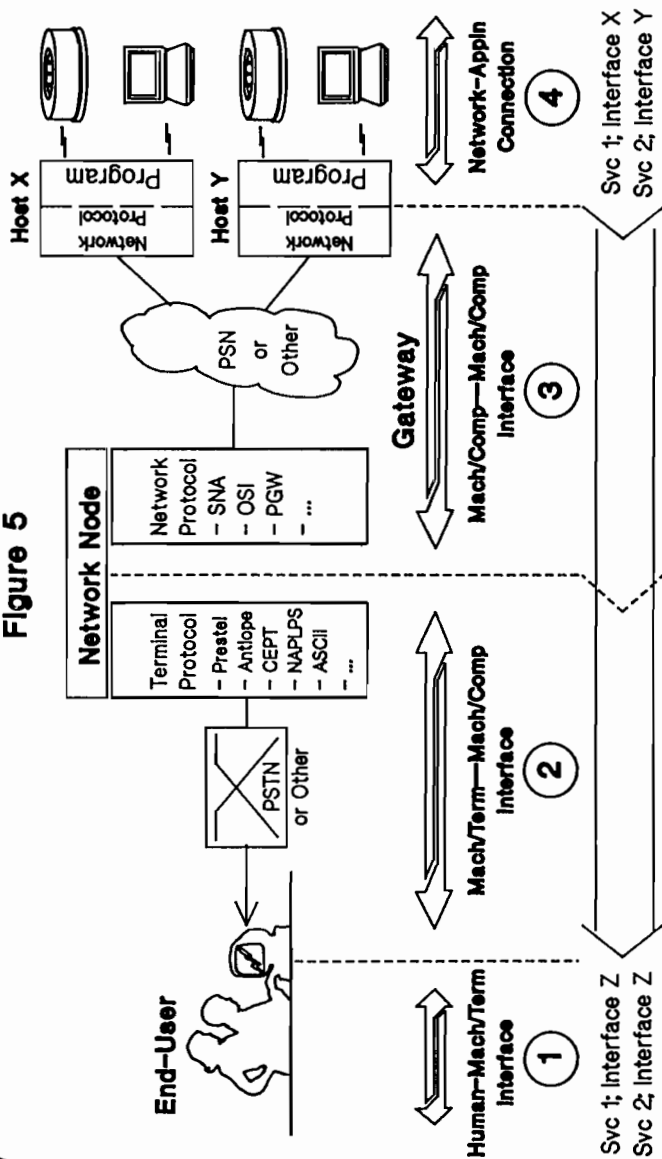


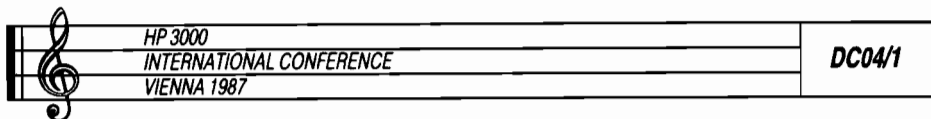


Figure 5



UNIFORM INTERFACE to DISTRIBUTED SERVICES (UIDS)

VICORP



# **NETWORK MAINTENANCE MANAGEMENT**

by

**Brian Button**

**Hewlett Packard Company**

## **CONTENTS**

### **ABSTRACT**

### **SITUATION ANALYSIS**

### **PRODUCT SUPPORT VERSUS NETWORK SUPPORT**


### **NEW ACTIVITIES AND RESPONSIBILITIES FOR MAINTENANCE MANAGEMENT**

SUPPORTING OTHER VENDORS' PRODUCTS  
SUBCONTRACTING CONTRACTUAL MAINTENANCE  
DIRECT SERVICE DELIVERY  
TECHNICAL EXPERTISE  
MANAGING A SUPPORTABLE NETWORK  
NETWORK TOPOLOGY MANAGEMENT

### **ATTRIBUTES OF NETWORK MAINTENANCE MANAGEMENT**

NETWORK SUPPORT OWNERSHIP BY SERVICE VENDOR  
FAULT ISOLATION  
DETAILED MAINTENANCE PLANNING  
NEW PRODUCT ELIGIBILITY  
NETWORK TOPOLOGY MANAGEMENT  
UPDATE MANAGEMENT SERVICE  
FLEXIBLE COST STRUCTURE  
PROBLEM REPORTING

### **TERMINOLOGY**

	HP 3000	DC04/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## ABSTRACT

To take advantage of the network capabilities available today, network customers frequently purchase products from several different vendors. This makes it difficult to arrange maintenance of the network. The original product vendors may not have appropriate support offerings. Also, eliciting cooperation between vendors for network problem solving can be difficult. The customer needs an entity which will take responsibility for the resolution of problems on the customer's network, when they occur. This is the objective of Network Maintenance Management, a service by which a single service vendor will manage a customer's entire network maintenance needs.

This paper explores the arena of network maintenance, specifically addressing Maintenance Management. It clarifies the difference between network support and product support. It then examines the responsibilities and activities required to deliver Network Maintenance Management and the implications the service holds for the relationship between the customer and the service vendor. Finally, based on the previous analysis, it reviews the probable attributes of a Network Maintenance Management service.

## SITUATION ANALYSIS

To take advantage of the network capabilities available today, network customers frequently purchase products from several different vendors. Very few vendors have quality product offerings which include all necessary types of data communications products (e.g. coaxial and standard cables, multiplexors, modems, X.25 switches, interface cards for processors, bridges, leased lines and public data networks). As a result, network customers typically purchase from more than one vendor.

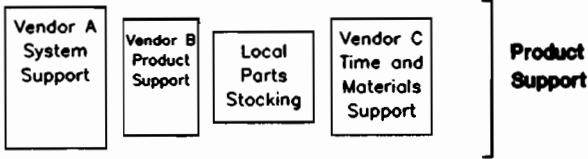
This makes it difficult for customers to arrange maintenance. Relying on the vendors from whom they purchased the products is frequently not a good alternative. Many vendors don't have an appropriate support offering. Moreover, having many vendors supporting part of the network can create problems. Each vendor has responsibility for their products rather than the operability of the overall network and will tend to disown problems which aren't clearly theirs. This tendency is known as "fingerpointing". A recent survey\* showed that 85% of network customers felt that fingerpointing was a problem.

Customers need an entity they can turn to which will dependably isolate and diagnose network problems. Moreover they need that entity to stay involved, coordinating resources, working with the customer and other vendors until the problem is resolved. Ideally, the entity should be a single point of contact and responsible for the resolution of problems on the customer's network, when they occur. This service, called Network Maintenance Management, is the subject of this paper.

\*International Data Corp., 1985 "Service in the Telecommunications Industry"

## PRODUCT SUPPORT VERSUS NETWORK SUPPORT

To clarify further discussion, it is worthwhile to examine the difference between network support and product support.

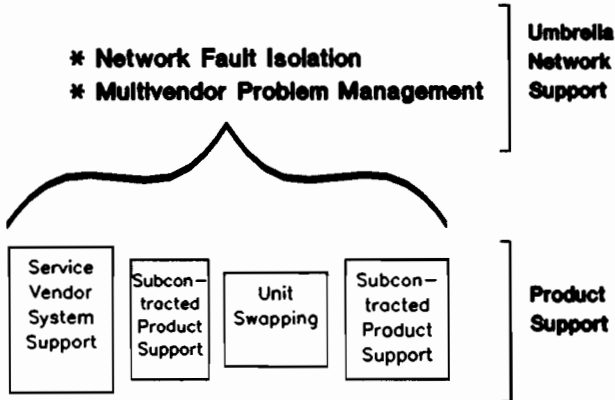


### Product Support on a Multivendor Network

The diagram above illustrates the situation where a customer has a network with many individual products being supported by different vendors. While each product on the network is supported, many problems still may be difficult to associate with a single product or vendor. Some examples:

- ▶ Products which don't work together or don't function in certain configurations
- ▶ Incorrect configurations or installation
- ▶ Improper cabling
- ▶ Intermittent software or hardware problems

These types of problems are common in networks. However, in the situation above, where the customer has product support from many vendors, these problems may cause added network downtime and customer frustration until the nature of the problem is determined and the appropriate vendor rectifies it.



### Network Under Maintenance Management

The diagram illustrates a network where the customer has Network Maintenance Management. Under Maintenance Management, the service vendor will isolate faults on the customer's network. Moreover, as the diagram indicates, support of the products is owned, directly or indirectly, by the service vendor.

When this customer has a communication problem, one entity, the network service vendor, is responsible for determining the origin and nature of the problem. The same vendor is responsible for resolving these problems.

## NEW ACTIVITIES AND RESPONSIBILITIES FOR MAINTENANCE MANAGEMENT

While the employment of a single service vendor will reduce customers maintenance worries considerably, it creates some new responsibilities for the service vendor. The purpose of this section is to examine these responsibilities and alternative ways of meeting them. Subsequent sections use the foundation created by this discussion to analyze the specific attributes of a Network Maintenance Management service product.

### SUPPORTING OTHER VENDORS' PRODUCTS

To manage support of the network, the service vendor must take ownership of the service of the individual products on the network, many of which the service vendor normally doesn't support. This may be done in one of several ways:

- ▶ directly delivering support on the other vendor's product
- ▶ subcontracting support to an existing support organization which specializes in supporting that product
- ▶ keeping a local supply of the product and swapping whole units in and out of the customer's network when they malfunction
- ▶ developing cooperative relationships with vendors where support ownership by the Maintenance Management service vendor is not feasible (e.g. PTT's, U.S. phone companies, X.25 service vendors, etc.)

Each one of these alternatives has its strengths and weaknesses and is likely to be applicable under different circumstances. Directly delivering support on other vendors' products has advantages because of the control and, for high volume products, the cost savings it allows. However, putting direct support in place is an expensive endeavor, requiring investment in special tools, training and stocking a parts pipeline. Hence, it is only an attractive alternative where volume levels are very high.

Subcontracting is likely to be an attractive alternative in many cases where support on the product already exists, especially with the wide variety and potential low volume of many network products. Subcontracting will frequently allow the service vendor to own support of other vendors' products much more quickly and less expensively than direct service delivery.

Sometimes, especially with low cost or unusual network components, support services will not already exist for the product and volumes won't allow the service vendor to gear up for supporting it. In this case, keeping local inventories of the product, either on the customer's site or at the service vendor's local office is the most reasonable alternative. Entire products can be swapped out when problems occur. This is only feasible for low cost product which are mostly hardware since software problems can't be resolved by swapping units.

Finally, in the case of major media vendors (phone companies or X.25 vendors) a joint maintenance relationship is the best alternative. Major media vendors provide a product and service on that product in one package, so subcontracting service from them doesn't apply the same way as with other types of products. Moreover, legal restrictions and the nature of most media vendors tends to eliminate subcontracting as an alternative. However, a Joint Maintenance relationship between the service vendor and the media vendor will significantly aid problem solving. The Maintenance Management service vendor will remain the first point of contact for communications problems. If, after some preliminary troubleshooting, evidence indicates the media vendor needs to become involved, the service vendor may directly contact the media vendor and act as the problem manager and technical interface for resolving the problem.

Two of these alternatives, subcontracting and direct service delivery are discussed in more detail below since they have strong impacts on the attributes of the Network Maintenance Management service.

## SUBCONTRACTING CONTRACTUAL MAINTENANCE

While the use of subcontracting considerably reduces delivery costs for support on third party products, it creates some new challenges. These are discussed briefly below.

- **Obtaining consistent availability.** One can't subcontract a service if the service is not available. If a customer's network is worldwide and support for one of the products on their network is not, the service vendor must develop some alternative methods for servicing the product in some areas. This will probably mean support delivery by the service vendor.
- **Providing consistent terms and conditions to the customer.** If several different vendors are subcontracted from, there is likely to be a variety of terms and conditions for the different products on the network. The service vendor must mesh many different coverage times, response times, geographic availability and costs into a single consistent service. This may be done by creating *classes* of service which encompass most of the types of support services. New vendors' services must be fit into one of the defined services.
- **Problem Tracking and Job Reporting.** Most service organizations have internal job tracking systems which allow them to track the status of customer problems. However, when multiple vendors are involved in managing a single problem, other coordination is necessary. The service vendor must make arrangements with its subcontractees so that the service vendor can monitor the status of problems being worked on by its subcontractees.

Similarly, subcontractees must be disciplined to use consistent reporting methods. This will allow the service vendor to monitor work done on specific problems, reconcile billing and determine subcontractee performance.

## DIRECT SERVICE DELIVERY

In cases where support on the product doesn't exist, is unsatisfactory or where product volume is high, the service vendor may decide to directly deliver support for a network product. This is typically not a quick process. Adopting support requires significant preparation, including:

- Establishing a relationship with the initial vendor as a dependable source of parts, training and any special purpose tools.
- Targetting individuals in the service vendor's organization for product training and training them.
- Setting up local and regional parts inventories.

## TECHNICAL EXPERTISE

This is probably the most important capability the service vendor must have. While delivering product support on other vendors' products requires good operational and administrative adjustments by the service vendor, delivering the umbrella fault isolation and properly managing other vendors requires an extra dimension of technical excellence. These responsibilities lie *directly* with the service vendor, and require high technical expertise for the following reasons.

- **The service vendor must isolate network problems in a generic network environment.** The members of the service vendor organization will be responsible for isolating faults in networks which have many products with which they are not familiar. They will not be able to rely on product specific knowledge but must have the generic datacommunications skills to determine the nature of the problem.
- **The service vendor must be able to guide other vendors through problem resolution.** In order to work with Joint Maintenance vendors and subcontracted service vendors the Maintenance Management service vendor must develop a reputation for making accurate problem diagnoses. It must also be able to muster sufficient technical data to convince skeptical Joint Maintenance or subcontracted vendors that its diagnosis is correct.

## MANAGING A SUPPORTABLE NETWORK

In the previous sections, we have discussed two important capabilities necessary for delivering Maintenance Management on a customer's network. Specifically:


- Being able to deliver product support on the products in the network
- Having the technical expertise to isolate network problems to their source on a network

These two responsibilities create a third new responsibility, namely:

*The service vendor must work with the customer on an ongoing basis to make sure the customer's network continues to be supportable and is technically sound.*

While it is the objective of Network Maintenance Management to provide support on a wide variety of customer's networks, supportable networks require planning. This must occur on an ongoing basis since networks tend to change almost continuously with the maintenance requirements of the network changing along with it. This consists of two parts,

- **Helping the customer to select products which are sound and can be supported.** The service vendor must be able to troubleshoot the network with the product on it and be able to arrange one of the service alternatives described above. As the network changes, the customer will tend to want to add new datacommunications products to the network. New products will require an evaluation by the service vendor. This is the subject of the section titled "NEW PRODUCT ELIGIBILITY".
- **The network must stay supportable as it is changed.** The service vendor, having committed to supporting the customer's network, must understand what it looks like and what products are on it. This is the subject of the next section, titled "NETWORK TOPOLOGY MANAGEMENT".

	HP 3000	<b>DC04/7</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## NETWORK TOPOLOGY MANAGEMENT\*

When a service vendor takes responsibility for the maintenance of the customer's network, a partnership is forged to manage changes and updates to the network. This is important to the service vendor because:

- Not all network configurations are supportable. The service vendor must understand the network in order to commit to supporting it.
- The service vendor is supporting the products on the network, some directly and some through other methods. This support requires planning by the customer and the service vendor. This planning is not possible unless the service vendor is involved in changes to the network.
- Troubleshooting is much easier on a well charted network.

This partnership is important to the customer because:

- Improper changes to networks cause a large percentage of the overall problems with a network. The service vendor can help the customer avoid network downtime by being involved in network change activity.
- The customer will receive regular copies of up-to-date, detailed information on the network. This information is invaluable for managing a network.
- The customer's network problems will be resolved more quickly because the service vendor's representatives will be able to start troubleshooting immediately without first documenting the network topology.

### TECHNOLOGY FOR NETWORK TOPOLOGY MANAGEMENT

The service vendor must have a software/hardware tool to assist with the management of the network topology. This application should have the following capabilities:

- A graphics interface for display of the network. It should be object oriented to allow moving up and down levels of detail in the network and retrieving information on specific devices.
- Capability of generating a graphic/textual hardcopy output for the customer and members of the service vendor's organization.
- Ergonomic, integrated graphic/textual interface for updating information on the network.

### METHODOLOGY FOR NETWORK TOPOLOGY MANAGEMENT

While a tool is important for managing the customer's network topology, a methodology for managing network changes is necessary to put the tool to best use. This is the central part of the customer/service vendor partnership. While many arrangements are possible, the basic requirement is that the service vendor must be involved in network changes.

---

\***TOPOLOGY** in this document refers to the layout or hardware configuration of the network.



## **ATTRIBUTES OF NETWORK MAINTENANCE MANAGEMENT**

### **NETWORK SUPPORT OWNERSHIP BY SERVICE VENDOR**

This is the primary objective of Network Maintenance Management. The service vendor, directly or indirectly, delivers support on the customer's entire network. In doing so, the service vendor takes responsibility for the proper operation of the network. The customer no longer has to track the levels of support on their network products, understand many different support organizations and, most importantly, manage these organizations when problems occur.

### **FAULT ISOLATION**

Fault isolation is a critical part of Network Maintenance Management, both from the point of view that it requires significant expertise in the service vendor's organization and that, once completed, problem resolution is usually a much easier step.

After this has been done, the service vendor can marshal resources to resolve the problem according to the pertinent product support arrangements on the network.

### **DETAILED MAINTENANCE PLANNING**

When a customer is considering Network Maintenance Management, he/she must work with the service vendor to plan for support. Some of the steps which must happen are:

- The service vendor needs to understand what the customer's network looks like and what the customer's needs are. Some of the questions which must be asked are:
  - Is this a planned network or is it in place?
  - What is the geographic range of the network?
  - What are the protocol types being used?
  - Are the products currently qualified by the vendor sufficient or are additional products needed?
  - Are there support solutions for all the network products in all of the geographic reaches of the network?
- If the customer needs new products added to the list of qualified products, the vendor must take time to qualify them. Qualification may take some time. If the service vendor is to do direct service delivery on the product, some additional time may be required to put this in place.
- There may be cases where the geographic distribution of the network or other factors require the service vendor to adopt some other support arrangement besides subcontracting or direct service. For example, it may be necessary to stock replacement units on the customer's site. These arrangements will have to be made jointly by the customer and service vendor, based on the circumstances.
- The service vendor will have to set the customer's expectations regarding the service being delivered. For example, a customer desiring 7 day, 24 hour response time will have to understand that perhaps not all of the subcontracted vendors on the network will respond to this need.

	HP 3000	<b>DC04/9</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## **NEW PRODUCT ELIGIBILITY**

The service vendor must be able to troubleshoot networks containing other vendor's products and must also be able to arrange for support of the products. This implies that the service vendor must maintain a list of third party products on which Network Maintenance Management can be delivered. This has the advantage for the customer that:

- The service vendor's field organization is trained and prepared to support networks containing that product.
- The products on the list have been tested and are technically sound.
- Workable configurations for that product have been predefined and the customer need not worry about compatibility problems.
- Support can be arranged for those products.

Customers will, naturally, want to add products to the service vendor's list. This will have to be a service available to customers as part of Network Maintenance Management.

## **NETWORK TOPOLOGY MANAGEMENT**

As mentioned in the previous section, Network Maintenance Management will create a partnership between the service vendor and the customer for managing the topology of the customer's network. Hence, topology management is closely tied to Network Maintenance Management.

Since the network and changes to it will be closely tracked by the service vendor, a likely alternative is for the vendor to provide a periodic report to the customer describing:

- The network topology (layout)
- The types, brands and configurations of devices on the network.
- Changes made to the network during the last period

## **UPDATE MANAGEMENT SERVICE**

To keep dependably accurate network information, it is important that the service vendor be apprised of or involved in network changes. If the customer uses an in-house staff or other organization to make updates to the network, there is a very good chance that the network documentation will become out of date. If the network documentation tends to become incorrect, it will become undependable for troubleshooting, and support planning. This will undermine the purpose of Network Topology Management.

One method of assuring that the network documentation is kept up to date is to have the service vendor make the changes to the network. This could be done as part of the Network Maintenance Management service for small to medium sized changes. For bigger changes or, perhaps, in all cases, a separate update service could be made available to customers.

### **FLEXIBLE COST STRUCTURE**

With Network Maintenance Management, unlike many other support services, it will be difficult to absolutely specify the product costs and deliverables until the customer's needs and network characteristics are determined. Until the customer's network is documented and support mechanisms for the customer's network products have been selected and priced, many cost variables exist.

While this makes it difficult to quote an absolute price for Network Maintenance Management, it is the responsibility of the service vendor to make its pricing and delivery as simple as possible. Subcontracted support terms should be arranged in predefined categories and multiple alternatives, if they exist, must be positioned clearly against each other.


Pricing for other alternatives must be clearly defined in formulas which account for geographic zones, product volume, product price and other pertinent variables. When these variables are determined, then the customer should be presented with an easily understood schedule. If changes occur to the network (see NETWORK TOPOLOGY MANAGEMENT above) then the pricing schedule should be redone, with changes highlighted.

### **PROBLEM REPORTING**

This would be a periodic summary of where the customer had trouble, what was fixed, how long it took and other, similar data.

This information will be important to the customer and the service vendor because:

- It will be necessary to justify per-incident charges, if any.
- It will help identify problem areas in the network.
- It will help monitor performance of subcontracted support vendors.

	HP 3000	DC04/11
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## TERMINOLOGY

*Hardware Configuration* - The description of the hardware devices in a network and how they are connected together. Also referred to as *topology* or *network topology*.

*Joint Maintenance* - The process whereby two vendors, having an previously established agreement, may contact each other and work directly together to resolve network problems which involve both of their products.

*Media Vendor* - Vendor of a media service such as a public data network (usually X.25) or a phone company.

*Network Maintenance Management* - A service by which a single service vendor will manage a customer's entire network maintenance needs.

*Service Vendor* - In this paper, *service vendor*, or *Maintenance Management service vendor*, consistently refers to the service vendor which is delivering Maintenance Management.

*Subcontracting* - In this paper, subcontracting refers to the purchase of product support by one service vendor from another.

*Topology* - The description of the hardware devices in a network and how they are connected together. Also referred to as *hardware configuration*.

*Unit Swapping* - A product support mechanism consisting of keeping spares close by (on the customer's site or at a local office) and swapping them into the network when the originals malfunction.



*HP 3000*

*INTERNATIONAL CONFERENCE*

*VIENNA 1987*



**MAKING EXTERNAL DATA AVAILABLE ON A MICRO**

Ronald W. Collison  
DARPA  
Arlington, Virginia  
USA

Users are starting to demand that data residing in centralized data bases and on other computers be made available to them on their micro computers without the user having to rekey the data. Additionally users want to make changes to data locally on their micros, then have the data transferred back and posted to the centralized data base or other computer. This paper looks a several ways to move data between two or more micros and micros and mainframes.

**TRANSFERRING DATA BETWEEN MICROS AND MAINFRAMES**

Moving data between mainframe/host computers and micro computers can be accomplished in several ways, including:

- 1) connecting the micros to the mainframe and using the mainframe as a file server;
- 2) soliciting the aid of a file transfer protocol and corresponding software to do a file transfer;
- 3) connecting to the mainframe as a terminal and using the data transfer capability that most terminal emulators offer;
- 4) extracting the information to be transferred and attach or embed

it in an electronic mail message;

- 5) contracting with one of the many commercial service organizations who specialize in converting information to micros;
- 6) choosing a more manual approach like printing it out then using an OCR to redigitize it.

### **TRANSFERRING DATA BETWEEN TWO OR MORE MICROS**

Moving data between two micro computers can be accomplished in several ways, including:

- 1) connecting the micros to another micro being used as a file server or connecting the micros to a specialized file server made especially to serve micros;
- 2) using a micro to micro file transfer protocol and corresponding software to do a file transfer;
- 3) direct connecting of two micros;
- 4) extracting the information to be transferred and attach or embed it in an electronic mail message;
- 5) contracting with one of the many commercial service organizations who specialize in converting information to micros;
- 6) choosing a more manual approach like printing it out then using an OCR to redigitize it.

### **FILE SERVER**

Several vendors now offer local area network approaches that allow the mainframe to function as a file server to a network of micros, including HP. Other vendors offer software that, 1) allows a micro computer to function as a file server to other micros or 2) works on a specialized server computer built especially as a micro file server. The file server approach to data transfer will likely be

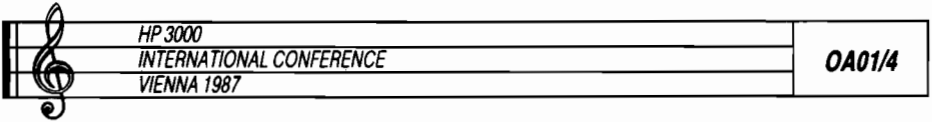
the easiest method of moving data between computers, when the file server supports automatic conversion of its file/data structure to that of the receiving computer. Under this approach, the user can run a query to extract the data he/she wants from the sending computer and store it on the server's disk. Then to make the data available to the receiving computer, the user need only reference the requested file as if it was generated on the micro.

HP's product, HP RESOURCE SHARING turns the HP3000 into a file server for connected HP and IBM/PC compatible micros. Extracted HP3000 data can then be stored on one of the HP3000 disks and later read and used by a connected micro computer.

#### **FILE TRANSFER**

When not using a common file server a good method for transferring data between computers is to perform a file transfer. File transfers use a common protocol for exchanging the data over telecommunications links. There are several protocols available for file transfer. One of the best known file transfer protocols for transferring data asynchronously between computers is KERMIT. KERMIT was developed at Columbia University and named after the Muppets cartoon character, Kermit the Frog. KERMIT is fast becoming a defacto standard for transferring files between computers. KERMIT allows the user to transfer files between micros and mainframes, micros and other micros, and mainframes and other mainframes over public and private telecommunications lines and direct asynchronous connections. KERMIT is widely available for most popular micros and mainframes (See partial list of hardware availability at the end of this paper). KERMIT's protocol guarantees a reliable file transfer.

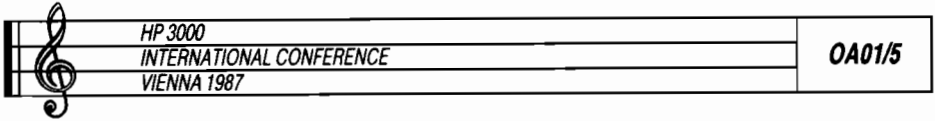




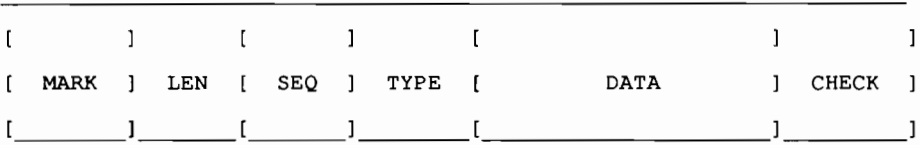
KERMIT also comes with a terminal emulator that supports TTY and VT100 terminal emulation and has multiple scrolling screen memory.

Our initial requirement was to move budget data from an HP3000 IMAGE data base to various micro computers. Since we primarily use IBM/PCs or compatibles, we were covered with KERMIT for the PC, but lacked a sister version for the HP. Since KERMIT source code and programmer instructions were available we were able to develop a version of KERMIT for the HP3000. KERMIT now allows us to make reliable file transfers of data between our HP3000s and our various micro computers, including HP's Vectra line of micro computers, IBM/PCs, Compaq micros, Gridcase micros, Macintosh micros, and SUN Microsystems workstations. We also use KERMIT to transfer data between the micros and our DEC/VAXs and DEC SYSTEM/20s.

KERMIT works by having two copies of the program running, one on the sending computer and one on the receiving computer. KERMIT is a character oriented protocol. Though KERMIT will send binary files, you have to identify them as such by invoking the SET command on both computers. KERMIT expects ASCII files to be transferred as the default. Data is transmitted between the two computers in packets. A check field is computed prior to data transmission, then recomputed by the receiving computer to verify that all data in the packet was received correctly. If the check field does not compute, an error is assumed and the receiving host requests that the sending host resend the data in the bad packet.




KERMIT PACKET LAYOUT (BASIC)



- MARK - REAL CONTROL CHARACTER, USUALLY CTRL-A
- LEN - LENGTH OF REMAINING PACKET
- SEQ - PACKET NUMBER SEQUENCE
- TYPE - PACKET TYPE
- DATA - DATA BEING TRANSMITTED
- CHECK - CHECK SUM USED TO VERIFY ACCURACY OF DATA RECEIPT

Columbia University has chosen to make all versions of KERMIT available to any requestor for a nominal copy charge for copies distributed on floppy disk/tape or free if picked up via one of the internetworks, like ARPANET or by making a copy of KERMIT from someone else. Object and source code is available. The only restrictions on its use are: 1) you can not sell it, 2) any enhancement or vendor implementations you make are to be made available to the user community either directly by you or by forwarding copies to Columbia University, and 3) Columbia University or the author of the KERMIT variation be credited with the creation of the underlying code.

Though KERMIT is essentially free, many KERMIT versions carry a copyright notice intended to protect the authors and sponsoring organizations from having their work turned into commercial

	HP 3000	OA01/6
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

products. Copyright notices generally give the user permission to change the source code and distribute freely both the object and source code as long as there is no charge other than a nominal administrative charge to cover copying and mailing costs and the copyright and author credits are retained in the copies given away. Also, vendors can use the KERMIT protocol in products that they sell as long as they do not charge extra for the inclusion of the protocol.

KERMIT is truly international. KERMIT is in use in countries around the world, including: both Chinas, East and West Germany, USSR, Israel, Iraq, New Zealand, New Caledonia, Chile, Mexico, Czechoslovakia, Malaysia, Sweden, Switzerland, Norway, and the Netherlands.

SEE SAMPLE HP3000 TO IBM/PC TRANSFER AND IBM/PC TO HP3000 TRANSFER AT THE END OF THIS PAPER.

#### **TERMINAL EMULATOR TRANSFER**

If you do not have access to KERMIT or another reliable file transfer protocol and supporting software and you have reliable communications lines, you could move the data between the computers via one of the popular terminal emulators that support data transfer. Most do not perform error checking and resending of bad data, though, so the user has to be more cognizance of what in being transferred and received. If your communications lines are reasonably reliable, you can probably safely transfer small quantities of non critical data. The problem is that if infrequent errors do get transferred, they are normally very difficult to

detect. Most of the popular HP26xx emulators provide this functionality.

### **ELECTRONIC MAIL**


Electronic mail offers still another alternative to move data. Many popular electronic mail systems allow you to attach a file for transfer with the message. When the user receives the file, all he/she need do is strip off the message heading and optional comment information and save the file. Note: moving files this way may limit the file record size and format.

In the future, wide adaptability of X.400 (the ISO mail protocol) will make the electronic mail transfer options even more attractive. Until then, users will be restricted to moving data within connected, compatible networks. Therefore, this option is only viable if the two computers are on the same network or a connected internetwork.

### **CONTRACTING**

Several vendors have started supporting converting data from one manufactures hardware to another's. This type of service is intended primarily for one time conversions or for infrequent use by those customers who neither have the time, expertise or volume to justify acquiring the tools and learning how do make the transfers themselves. The service is frequently costly and limiting.

MCI Communications Corporation and LOTUS Development Corporation have recently signed a pact to offer a combination product/service for moving data between mainframes and micros and between two or

	HP 3000	OA01/8
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

more micros for those customers desiring to move data to remote sites. The data transfer product/service will transfer any piece of data from one program to an identical program on the receiving end. Other vendors have also announced their intent to offer similar services, including Western Union.

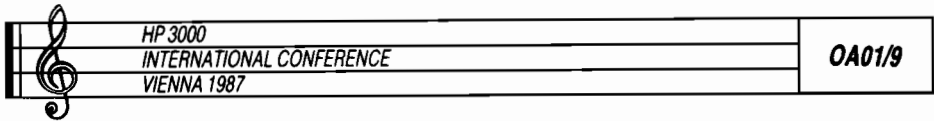
#### **OPTICAL CHARACTER RECOGNITION (OCR)**

Use of an OCR is probably only one step better than rekeying the data, but if you have access to the OCR hardware it could be more efficient than retyping it.

#### **GETTING DATA LOADED DIRECTLY INTO AN APPLICATION**

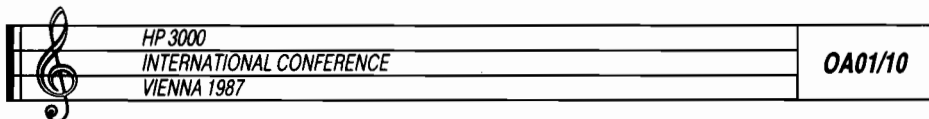
Many popular micro software packages provide for importing of foreign data from other applications via the DIF standard for data transfer.

With the introduction of the IBM/PC and our completion of an HP26xx terminal emulator for it, we had to make available an easy procedure that let the users of micro applications retrieve data from the HP3000 stored in an IMAGE data base and optionally return the updated data for data base update when finished changing it. Since we were trying to do the data transfer before commercial vendors started bringing products to the market to assist with data transfer between two or more micros or mainframes and micros, we had to invent a procedure for making the transfer of data.



The first application was one of extracting budget planning information from an IMAGE data base and building an electronic spreadsheet using the product LOTUS 123. To fulfill this requirement, we were able to successfully obtain information from LOTUS Development Corporation on their LOTUS 123 internal storage format from which we developed a C program to read data from a flat file and write a 123 spreadsheet or read a 123 spreadsheet and generate a flat file. With this tool, we could then have data extracted from the IMAGE data base and stored in a flat file in the format of the desired spreadsheet, which in turn would be read by the C routines and output a 123 spreadsheet in the correct format that looked like it was originally written by LOTUS 123. The upload of the spreadsheet resulted in a flat file that could be fed into the IMAGE query language we developed when IMAGE was first introduced called AQ. AQ's update capability would then process the flat file like individually keyed records and proceed to update the data base.

Problems encountered in the upload function were primarily centered around how to control the data quality of the uploaded spreadsheet. Instead of a few trained data entry persons keying the data or a transaction processor, processing the users input and flagging errors as they are entered, we now had the possibility of many users being able to play games with their budget data, then attempt to have the data automatically applied to the central data base residing on an HP3000 using the IMAGE DBMS without adequate data quality and access permission controls.



Recently there have been many new products introduced to make moving data from mainframes to micros and between two or more micros easier, including HP's new products; HP RESOURCE SHARING AND INFORMATION ACCESS. HP RESOURCE SHARING allows users to use the HP3000 disks as file servers for PCs connected to the HP3000 LAN. HP INFORMATION ACCESS is an enhanced version of HP ACCESS, which allows the user to transfer data from an HP data base or file into a PC application.

#### **KERMIT AVAILABILITY**

##### **FOR THE HP3000**

KERMIT for the HP is available by contacting:

Ronald Collison

DARPA

1400 Wilson Blvd.

Arlington, Virginia 22209-2308

USA

##### **FOR ALL COMPUTERS**

General KERMIT information and various versions of KERMIT for micros are available by contacting:

Kermit Distribution

Columbia University Center for Computing Activities

612 West 115th Street

New York, New York 10025

USA

**KERMIT RUNS ON**

Kermit is available on over 200 computers including (partial list):

**MAINFRAMES**

Hewlett-Packard 3000s and 1000s

DEC PDP-11, VAX, DEC10, DEC20

IBM/370 VM/CMS, MVS/TSO, MVS/GUTS, MTS, MUSIC

PRIME

Tandem

Data General

Cray

Burroughs

Univac-1100

**MICROS**

IBM/PC and compatibles, including Vectras

Apple Macintosh

Apple II

Apollo

Atari

Commodore 64 & Amiga

DEC Pro-300

Radio Shack TRS80

Sun



**\*\*\* SAMPLE \*\*\***

**FILE EXCHANGE BETWEEN HP3000 KERMIT AND PC KERMIT**

**UP LOADING A PC FILE TO THE HP**

```

+-----+
| 1) First Logon to the HP3000 |
+-----+
:hello mgr.user
ENTER ACCOUNT PASSWORD:
ENTER USER PASSWORD:
HP3000 / MPE V G.01.06 (BASE G.01.06). MON, JAN 20, 1987, 3:20 PM
Welcome to the DARPA HP3000
+-----+
| 2) Then run the KERMIT3000 program |
+-----+
:kermit
HP 3000 KERMIT version 1.1A
+-----+
| 3) Next Put the HP3000 in server mode |
+-----+
KERMIT3000>server
+-----+
| After entering SERVER mode - escape back to your PC KERMIT |
+-----+

```

----- NOTE -----

| After putting the HP3000 in SERVER mode, depress the escape |  
| key to return control back to the PC and the PC version of |  
| KERMIT. The user is now using the PC and the PC version of |  
| KERMIT to issue the file transfer commands. Since |  
| Kermit3000 is running as a server, the PC will initiate all |  
| file transfer and directory management commands. |

| In this example, the user sends a file named "VDTE.CMD" |  
| from an IBM PC running Kermit-MS ver. 2.29, (the example |  
| actually sends the file to the HP3000 twice to demonstrate |  
| the file collision avoidance features. |

-----  
-----  
| 4) After sending the two files, the user issues the EXIT |  
command to stop the KERIT3000 server mode on the HP3000

KERMIT3000>exit

END OF PROGRAM

----- NOTE -----  
LISTF example of the files that were sent to the HP3000

:listf ,2

ACCOUNT= USER            GROUP= PUB

FILENAME	CODE	-----LOGICAL RECORD-----					----SPACE----		
		SIZE	TYP	EOF	LIMIT	R/B	SECTORS	#X	MX
TESTFILE		80B	FA	22	22	3	9	1	1
VDTECMD		80B	FA	46	46	3	17	1	1
VDTECM01		80B	FA	46	46	3	17	1	1

----- NOTE -----  
 | In the above example, the two files named VDTECMD and |  
 | VDTECM01 demonstrate collision avoidance, the second file was |  
renamed to prevent the first file from being overwritten.

**DOWN LOADING AN HP FILE TO A PC**

-----  
1) Logon to the HP3000 if not already logged on

-----  
2) Run the KERMIT3000 program

:kermit

HP 3000 KERMIT version 1.1A

+-----+

| 3) Initiate the transfer from the HP: |

+-----+

KERMIT3000>send testfile

+-----+

| 4) Escape back to your local PC KERMIT and put the PC into |  
| receive mode by entering the RECEIVE command |

+-----+

+----- NOTE -----+

| After the file is transferred, the operator returns to terminal |  
| (CONNECT) mode and EXITS the program |

+-----+

KERMIT3000>EXIT

**AVAILABLE KERMIT COMMANDS**

KERMIT3000>help

**KERMIT COMMAND SUMMARY:**

TAKE filespec

SERVER

SEND filespec1 [filespec2 (renames filespec1)]

FINISH

RECEIVE [filespec]

SHOW parameter (anything that can be set)

    SHOW ALL

REMOTE TYPE filespec

REMOTE DIRECTORY [filespec]

REMOTE SPACE [filespec]

REMOTE DELETE filespec

REMOTE DIRECTORY FOO@

EXIT

SET PARITY option (NONE MARK EVEN ODD)

SET DEBUG number

SET LOG filespec

SET HANDSHAKE option (XON ,NONE ,XON2)


SET LINE ldev

SET SEND PAUSE number

SET SPEED speed

SET SEND BINARY option (ON,OFF,AUTO)

SET DELAY number

	HP 3000	<b>OA01/17</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

SET RECEIVE BINARY option (ON, OFF)  
 SET RECEIVE DEVICE [dev]  
 SET RECEIVE FCODE n  
 SET RECEIVE RECLEN [-]n  
 SET RECEIVE BLOCKF n  
 SET RECEIVE FIXREC option (ON, OFF)  
 SET RECEIVE MAXREC n  
 SET RECEIVE MAXEXT n  
 SET RECEIVE SAVESP option (ON, OFF)  
 SET RECEIVE PROG  
 SET RECEIVE BIN128  
 SET RECEIVE TEXT  
 SET RECEIVE TXT80

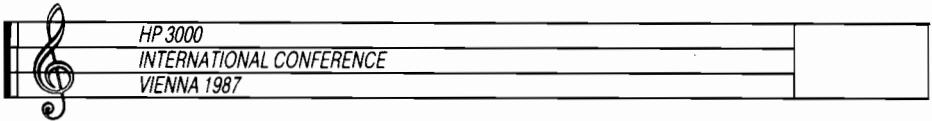


**Trademarks:**

IMAGE, HP RESOURCE SHARING, AND INFORMATION ACCESS are trademarks of the Hewlett Packard Corporation  
 LOTUS 123 is a trademark of the LOTUS Development Corporation  
 IBM/PC and PC are trademarks of the International Business Machines Corporation

**References:**

da Cruz, Frank, "KERMIT, A FILE TRANSFER PROTOCOL," Digital Press - Digital Equipment Corporation, 1987



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987



**Roni Klimesheffskij**  
**Posts and telecommunications of Finland**  
**Data Processing Centre**  
**29A Elimaki st.**  
**00511 Heslunki 51**  
**Finland**

## Software Super Market

- Software Super Market briefly
- Users to support
- Data Processing Centre
- Software Super Market - system

## The Gateway project

- The objective
- The starting point
- Presenting the three parts
- Basic results of the experiment
- How to proceed



## Software Super Market

### Software Super Market briefly

Software Super Market is a self-service shop, where customers (end-users) "buy" software to meet their needs. He or she may study a brochure of the product or test it in practise. Software Super Market is also the one place to start all general-information packages in the PTT internal data-network.

### Users to support

#### ADP- users in the PTT

With almost 46.000 workers to serve, out of which say 2000 are in some way HP- users, the Data Processing Centre's two HP- customer support people have a tough job. The every day support has been delegated to 15 system managers and local operators but they have to be supported too. In the General Directorate the HP's are used mostly to gather and analyze data entered in the regional and local administration. Data bases are huge and jobs are heavy. In the regional administration the HP's are used to run data-entry applications and to maintain local data-bases. With more and more office-automation type software available to be run in the same workstation as the old and new systems, the poor users would be totally lost without a radically new support system.



Software Super Market

Organizational chart

General directorate of posts and telecommunications			
<b>PTT OF FINLAND</b>			
<b>POSTS</b>	<b>FINANCE</b>	<b>ADMINIST- RATION</b>	<b>TELECOMMUNI- CATIONS</b>
Postal dept.	Finance dept.	Administ- rative dept.	Telecom- munications dept.
Postal traffic dept.	Materials dept		Telecom- munications technique dept.
			Radio dept.

Regional Administration

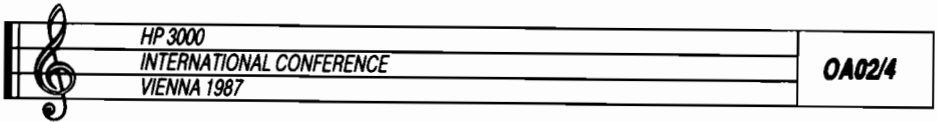
Postal service:  Nine postal districts	Telecom- munications service:  Eleven telecom- munications districts
-------------------------------------------------------	-------------------------------------------------------------------------------------------

Special Units

Philatelic centre.
Telecommunications laboratories
Data Processing Centre
.....

Local Administration

Postal areas	Telearea offices,
	Radio stations



## Software Super Market

### **Data Processing Centre**

#### **Supporting ADP- users in the PTT**

The Customer Support department in the Data Processing Centre concentrates on what we call personal workstation services, everything that provides the user with necessary and adequate tools for doing anything else than running systems. It consists of an info-centre to support the IBM- users, a micro- group, a HP- group and a group for organizing courses. ADP- systems have their own user support in the Systems development department.

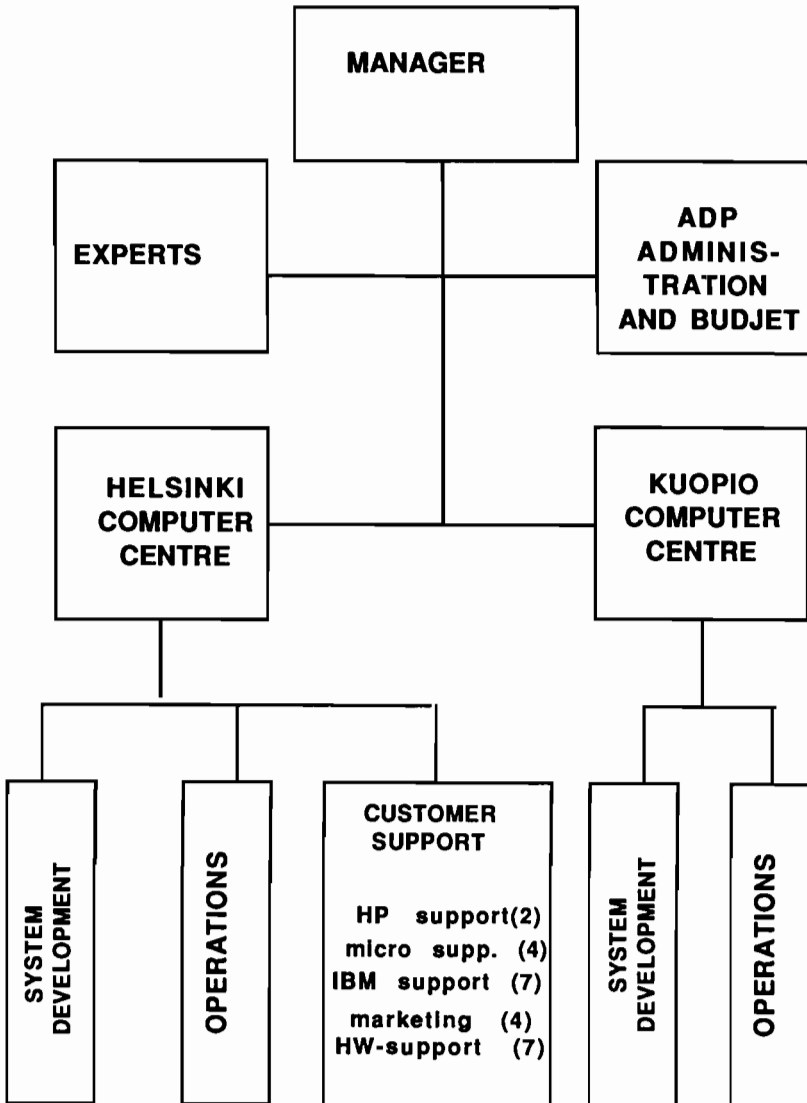


Software Super Market

Organizational chart

POSTS- AND TELECOMMUNICATIONS  
OF FINLAND

DATA PROCESSING CENTRE



## Software Super Market

### Software Super Market - system

#### Main objectives

The main objective is to provide all users in PTT's HP-network with a service to get information and test all HP-software which is supported by the Customer Support section of the Data Processing Centre.

Software Super Market also provides a gateway to all information services in the internal PTT data-network.

It also is a model of how user-interface should be implemented in all HP-workstations

#### Who is it ment for

Software Super market is ment for users, who are not bound to one or few adp-systemapplications only. It is ment to users who use their terminal as a personal workstation.

#### New users

New users in the PTT HP3000-network, who need a clear overview of personal workstation services

#### Users with specific needs

Users, who need to know what services are provided for a specific need.

#### Users of the PTT's general information services.

Now the users of PTT's HP-network can select general information to meet their needs from one menu in the Software Super Market. As present these services are electronic library systems for 12 libraries, a reference system for ADP-systems in the HP-network, a reference system for program documênts, a reference system for public personal files etc. The system managers in the regions have added some local information services to their Super Markests.

#### The benefits to HP-support

The HP-support group can now concentrate on consulting how to implement a new service to the customers. Before they had to spend their time explaining how the products work. Due to this new information channel users are better informed about new products, when they have learned to use the Software Super Market information service.



## Software Super Market

### How to get there

All the HP3000- computers know what the UDC: SUPERM - means. It initiates the local Super Market with local information and news. One choice of the menu is a gateway to Data Processing Centre's Super Market. System application menus can also be linked to the local Super Market, which provides useful additional services like terminal messageing and personal card-files for example and always in the same standardized way.

### The departments in Super Market

Software Super Market has three departments:

#### Information services department

All users of the Software Super Market are provided with a gateway to all PTT:s internal information services e.g. distributed electronic library service, application reference- service.

#### News department

This is the main information channel used by HP- support group when adding new products to the list of Supported Products. That always indicates that an information bulletin and a testig environment has been added to the Software Super Market. News on new versions and found errors are also provided via this channel.

#### Software bulletin department

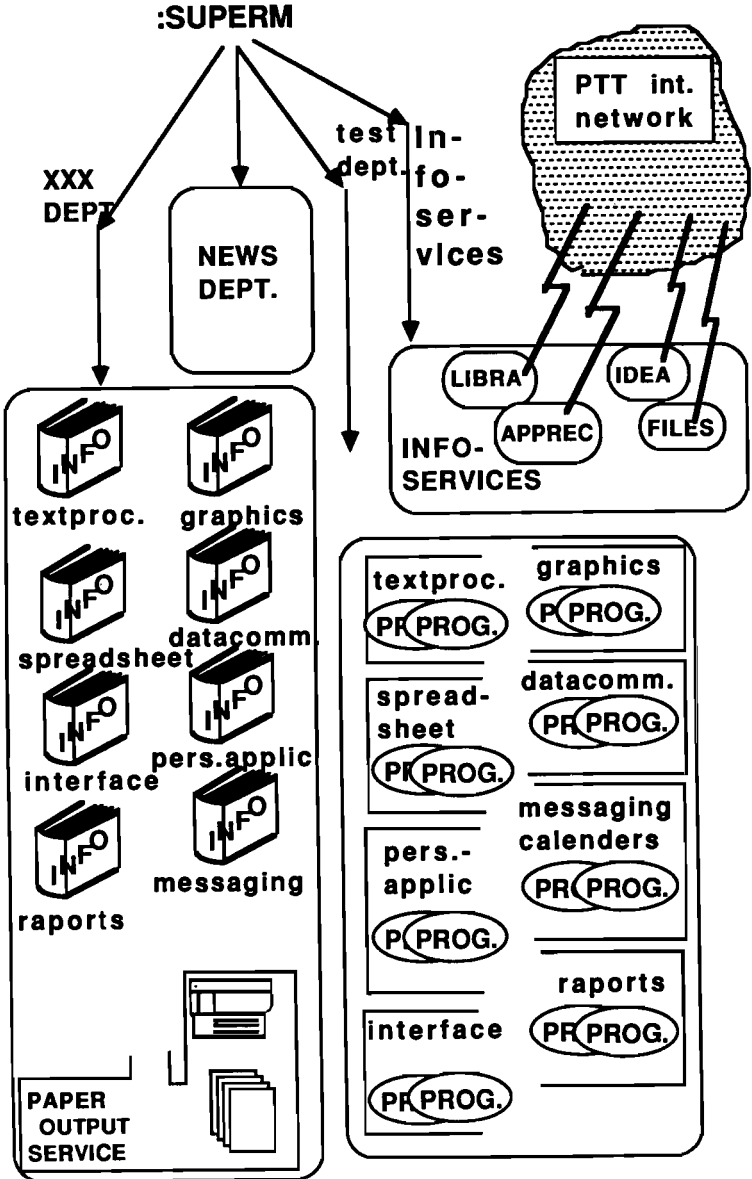
This department provides the customer with four to eight pages of information about each software product in the Super Market. The information is available locally in TDP- text processing format.

#### Testing department

In this department the customer can test all the software products in a demo environment with demonstration databases. If the product it self does not use function keyes, the keyes are provided by the Super Market to make sure that the user interface is as alike as possible in all different products.



Software Super Market  
Software Super Market layout



**How is it built**

Software Super Market -system is built using the MENUPROCESSOR (from TYMLABS) for processing menus and initiating tasks and TERMKEYS (from PTT) for handling

## Software Super Market

functionkeys and recognizing terminaltypes. The system is menu driven so that the user can select from the main menu submenus, tasks, streams or run programs. Tasks consist of MENUPROCESSOR- or MPE- commands. Remote Hello- commands in tasks and logon udc:s are used to get services from other computers in the DS- network. MENUPROCESSOR provides an on-line help facility for each service that is selected from the menu.

### **How is it distributed**

Each HP3000 in telecommunications districts has a local Super Market, which contains local information and services. One service in the distributed Super Markets is to enter the Data Processing Centre's Super Market. Users in the General Directorate use DPC's Super Market.

### **Benefits to users**

#### Building personal menus

It is very simple to build personal menus to individual users or usergroups. The menus can be linked together hierarchically in a tree format. New programs and services can be hooked easily to these menus.

#### To ease initiating applications

An application or ADP- service can be initiated by one selection only, no matter where it is in the PTT- internal network. It may be in any of the 19 HP3000-computers or in our IBM- mainframe. Software Super Market takes care also of strange filedefinitions and peripherals. Terminaltypes are also identified and thus the terminal capabilities can be used efficiently. Now at last the services can be called using their functional names.

#### To reach ADP departments' HP- support

Support people can seldomly be reached with telephone and only some of the customers who need help are HPDESKMANAGER- users. Now the support people can use the NEWS department to tell about new products and users have the basic support right at their terminals

#### To standardize information retrieval

Information services are gathered to one place and most of them use our general retrieval system, so that the interface is the same in all these applications.



## Software Super Market

### To standardize on-line help

Every application and service has a bulletin, which tells at least the name, function, general operating instructions and where to get more documents.

### To standardize text processing

The Super Market uses ready TDP- templates with standard margins and headers.

### To standardize user interface

All products are initiated in the same way and have standard function keys, e.g. F1 stands for help and F8 for exit.



## The Gateway project

### The Gateway project

#### The objective

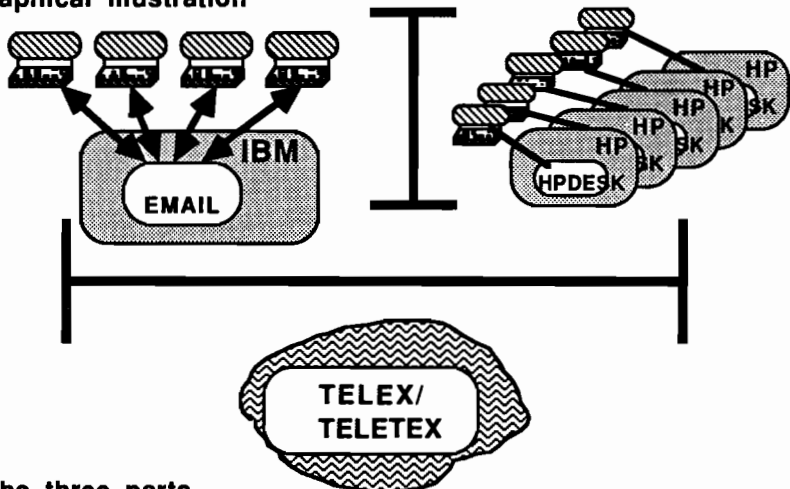
The objective of the project was to introduce a message- and textswitching facility between the three worlds in PTT's data processing environment, namely between the IBM, HP and telex/teletex worlds, so that any user of the three different message switching services can send a message to any other user.

#### The starting point

##### Three isolated message switching environments

HPDESKMANAGER enables messages to be sent across the HP3000-computer network in a very userfriendly way. Linking the users in 15 different HP3000-systems, so that they can send messages to one another is not a major problem. The problems arose when both HPDESK in HP and EMAIL in IBM were used long enough that they were "a way of living". It could not be accepted that a part of the Data Processing Centres' workers could not be reached. Another intolerable fact was the shortage of high quality printers. There were many teletex workstations all around the PTT. The teletex workstation would have been a very neat solution to the qualityprinter problem, if only it would have been possible to send texts from HP- terminals to teletex workstations.

##### A graphical illustration



### Presenting the three parts

#### Integrating HP3000 to teletex

The integration has been achieved by TTX/3000- program from KAAKONTIETO-company in Finland. The program has an user interface for sending telex- and teletex-messages from HP-asynchronous terminals. Receiving personal messages from teletex/telex network demands the integration to HPDESKMANAGER, which would take care of the routing. The

## The Gateway project

HP3000 is connected to the teletex-network via a Nokia teletex workstation. The Nokia workstation can be used as a normal teletex device in spite of the fact that it is the link between HP3000 and teletex. If used also as a normal teletex workstation, all in-coming messages must be manually transferred from the teletex network to HP3000. The TTX/3000 -program uses EDITOR/3000 or HPSLATE as its textprocessor. HPSLATE guarantees that text processing has an interface friendly enough.

### **Integrating HPDESK In HP to EMAIL In IBM**

The software to solve this problem is home maid, namely in the Data Processing Centre. The solution is based on two packages, JOBGEN/3000, which is represented in Finland by PORASTO, for constructing and timing jobs and IMAS/3000, by FJERNDATA in Norway, for IBM- terminal emulation and for transferring files both ways. The solution provides HPDESK- users in HP and EMAIL- users in ROSCOE the following advantages: 1) Whenever entering ROSCOE, a user gets to know how many messages he or she has received in EMAIL and in HPDESK. 2) Whenever entering the HP/3000, a user gets the same information as above. Both these notices can also be received at request. 3) Every HP- user can be provided with a short cut for entering IBM/ROSCOE/EMAIL with only one keystroke. All logons and passwords are naturally private.

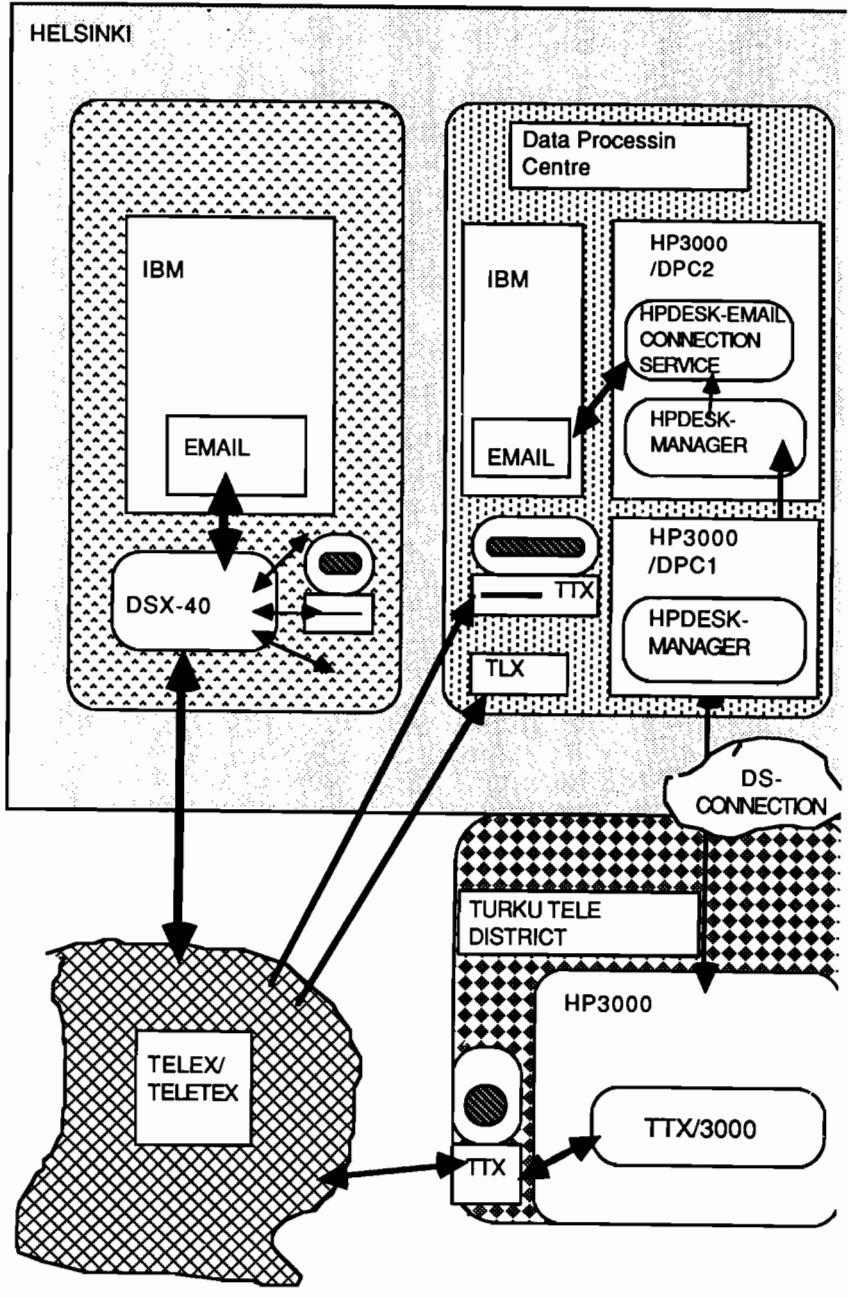
### **Integrating EMAIL to telex**

We used Philips DSX-40 telex equipment for this purpose.



### The Gateway project

Illustrating the test configuration



## The Gateway project

### Basic results of the experiment

#### HP3000 - teletex

The program was used for sending telexes and teletexes by many people, many more than originally planned. The only major complaint for sending messages was that it takes such a long time before the user is able to check if the message was sent successfully. We had some experiments for receiving messages too, but we found that receiving teletex- and telex messages is well organized everywhere. So that if a teletex message cannot be received in HPDESKMANAGER, there is no use in changing the present procedure where messages are printed in paper. The usage as a quality printer was a success, but this feature is useful only to those, who work close to the teletex workstation. Now, when the era of cheap laserprinters has begun, the importance of teletex workstations as high quality printers is rapidly decreasing. The user interface of TTX/3000 was clear and easy to learn.


#### HPDESK - EMAIL

It was hard to find test persons for this test, because people are so used to their own messaging environment that messages across the IBM-HP border were quite few. The secretarial workers found this service most useful, they have to contact many people regardless of the borders. This service serves mostly HP-terminal users and users with micros because the user should be able to enter any of the three message switching systems with his own workstation. We do not have software to enable HP-sessions to synchronous terminals. The secretaries claimed that this service of providing notices of received messages is enough, they found no need for converting all the messages automatically from EMAIL to HPDESK and vice-versa. This is strongly based on the fact that they have an automatic and fast entering procedure in their micros to both EMAIL and HPDESK.

#### How to proceed

The two major tasks that remain in linking the HP/3000 to teletex are replacing the present teletex workstation with an automatic adapter, which is dedicated to HP/3000 traffic and integrating the TTX/3000 package with HPDESKMANAGER.

In message switching we are moving towards OSI X.400- standard as an overall document- and message switching- standard. This means that all self maid connections between two messaging systems have hopefully a short life.

	HP 3000	OA03/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The Mini and the Micro  
Distributed Application Development and Processing

Karen Heater  
InfoCentre Ltd.  
6303 Airport Road  
Suite 300  
Mississauga, Ontario  
Canada L4V 1R8

Introduction

The role of the Personal Computer in the HP3000 data processing installation es due for some changes. In many cases the PC is underutilized, serving as a standalone workstation for word processing, graphics, and spreadsheet analysis. These are excellent uses for the PC, and remove the need for the aforementioned services to be offered on the HP3000, however we are now in the position to fully exploit the capabilities of our PC's and further reduce the load on the HP3000 associated with application system development and execution.

With new 4TH Generation development software and data communication technology, our Personal Computers can play an integral role in system development and distributed system processing. An associated challenge concerns maintaining the security of our distributed corporate data.

We will be describing in this paper, opportunities available for integrating the use of your PC's into your everyday application system processing, and we will be considering the impact these opportunities may have on the continued security of your corporate data.


Before launching into this discussion let's look at some justifications for expanding the use of our micros.

- 1) Cost. The cost of PC hardware continues to fall. It is not difficult to acquire an MS-DOS based machine, with a generous configuration for a purchase price less than or equal to that of a HP video display terminal. The cost of software is another consideration. There can be no question that PC software is available for a fraction of the price of mini-computer software having similar functionality.
- 2) Redundancy. What happens when your HP3000 is unavailable for use? Few HP3000 installations have a spare machine that can be pressed into service when catastrophe strikes. On the other hand, it is far more likely that an installation would have a spare PC on hand, to keep the micro based applications running when a particular machine is down.
- 3) Performance. There is a lot of computing power in our Personal Computers waiting to be harnessed effectively. This computing power can be put to work doing some of the tasks currently undertaken by our HP3000s. If the workload is distributed wisely, then optimal use can be made of both resources. Our overworked HP3000 can shed some of its burden, possibly stalling an impending upgrade. As a result, we can improve the performance of our mini, which in turn can make our user community more productive, and make better use of our available resources.

OK, so we know we should be making better use of our PCs, removing some processing burden from our HP3000's, but we need to equip our Data Processing Department with the appropriate tools in order to accomplish this integration.

Firstly we require an application development environment that is common between the two machines. Specifically, we require the same programming language on the micro as we have on the HP3000, and the same DBMS. It is advantageous for this common programming language to be a Fourth Generation Language (4GL). A 4GL offers several benefits critical to the successful integration of minis with micros:

- Portability. If the micro version of the 4GL is a true implementation of its HP3000 based counterpart, then applications developed using that language will be easily ported from one environment to the other. Furthermore, there is no need for the conversion of source to executable code (compile and link process) required by third generation languages, which again simplifies the porting process.

	HP 3000	OA03/3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

- Standardization. 4GL's standardize the appearance and behaviour of menus and screens, enabling users to work with different applications within the organization, with minimal application specific training. You can take this standardization one step further, making the micro screens look and operate like their HP3000 counterparts. Additionally, if the 4GL includes a module which generates user documentation from the source code, then this standardization benefit will extend to your system documentation.
- Speed of development. The same benefits being realized in HP3000 system development efforts using 4GL, directly apply to the micro environment.

A common application development environment provides two very important benefits:


- our system development staff can develop applications for the PC's with no new learning. All existing knowledge of the programming language and experience with creating, manipulating, and accessing Database files is transferable.
- application development activities can be undertaken either on the PC or the HP3000, regardless of where the finished product will ultimately run.

Secondly we require a mini to micro communication facility. This facility must enable communication between the machines in both directions. It will be used to transfer text files during the system development phase, and to transfer data when our users are running the application.

Consider as an example of these tools, InfoCentre's successful 4GL Speedware and associated micro based product microSpeedware. The combination of Speedware and microSpeedware provides a common programming language - REACTOR, while microSpeedware's Speedbase is an IMAGE alone for the MS-DOS environment. We will discuss later in this paper the communication facility that is available to the Speedware installation.

With these tools in place, let's turn our attention to integrating our micros with our minis. Like anything else, a good systems integration tool will provide you with choices, rather than locking you into one fixed "solution".



	HP 3000	OA03/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

For discussion purposes let's identify three approaches available to the Speedware installation for implementing micro-mini integrated solutions:

- 1) Standalone applications.
- 2) Batch Integrated.
- 3) OLRT Integrated.

The first approach entails identifying the machine for which a particular application is best suited, then developing the application to run only on that machine. This is what most of us are doing now. Our serious applications are developed to run on our HP3000's, frequently with little thought given to the role that our Personal Computers can play in the corporate systems strategy.

Consider however that some applications are ideally suited to run in a standalone fashion on a Personal Computer. Many shops have several candidate applications hiding in their applications backlog. These may be those system requests for a small standalone system, benefitting one department within the organization, that may not justify the allocation of HP3000 computing and development resources. Complicating matters further, the application may have several twists to it, making it unsuitable for the popular PC Database packages.

Given the toolset described earlier, the following scenario becomes possible:

Using an IMAGE application generator, such as the DESIGNER module of Speedware, develop the application on your HP3000.

Your system development staff are already familiar with the tools - Speedware and IMAGE, and the tools get the job done very quickly. An experienced Speedware user will produce results in a fraction of the time required by his COBOL oriented counterpart.

DESIGNER will generate the application, which consists basically of two components:

- an IMAGE Database
- a REACTOR specifications file containing the Speedware code for the Menus, Screens, Online HELP, Reports, and Transaction Processing programs required by the application.

The developed application can be implemented on the Personal Computer by downloading the IMAGE schema, and the Speedware code (Specifications file). This text file transfer can be accomplished with the use of the file transfer utility of your choice. The IMAGE schema becomes a Speedbase Database by compiling the schema text with the Speedbase Schema Processor. With the Database created on the PC, the user accesses the application by running microREACTOR against the downloaded specifications file.

Developing standalone PC applications as outlined above yields several benefits:


- the user gets the application he needs.
- you didn't add another application on to the load of your HP3000.
- your staff developed the application without embarking on yet another learning curve.

Ongoing maintenance to the application can be undertaken in the same fashion. Use DESIGNER to make programming or Database structure changes to the application, then download the new version (as described above). Alternatively, the application can be maintained locally (on the PC) using text editing software to implement programming changes to the Specification file, or structural changes to the schema text file.

To summarize on this standalone approach, the tools described above enable one to develop an application on either the Personal Computer or the HP3000, and then implement that application on either machine. This is made possible by the system development environment which provides a common DBMS and programming language on both machines.

In the example outlined above, an application was developed on the HP3000 for use on a Personal Computer. Once implemented, this application will run standalone on the PC, and that is where the data resides. Prudent computer system operation procedures dictate that the data should be protected and secured. The data can be protected by the implementation of rigorous backup procedures. This will be the responsibility of the PC user who should be encouraged to develop and practise these procedures. Data security poses a much bigger problem. In the absence of the familiar MPE/IMAGE umbrella, how do we prevent unauthorized access to the data resident on the Personal Computer? This problem introduces a major stumbling block which possibly limits the usefulness of this mini - micro integration approach to casual applications processing insensitive, non-critical data.

The second choice was labelled Batch Integrated. This approach enables an application system to be designed where the processing, and the data, is shared between the HP3000 and any number of

	HP 3000	OA03/6
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Personal Computers. With this approach, the communication between the HP3000 and the micros is batched. For example, at the end of the day, or the end of the week, all of the transactions processed by the PC workstations are uploaded to the HP3000 and posted to the central IMAGE Database. At the same time, new versions of the "master" or reference type Datasets are downloaded to the PC workstations, enabling them to carry on with the next batch of transaction entry.

Capitalizing on the common development tools in the two machine environments, this type of application can be developed and implemented quite easily. The micro to mini communication (the batch transfer of files) could be undertaken with a file transfer utility such as HP's AdvanceLink.

The advantages to this approach can be:

- the processing involved in data editing and general transaction entry is offloaded from the HP3000. The PC earns its keep.
- for remote workstations data communications costs can be reduced. The workstation is not connected all day, and the data transferred has been pre-edited by the application programs.

There are several drawbacks to this approach:

- information is not shared in a timely fashion. Depending on the application, this drawback will vary in severity.
- the duplication of data is costly, time consuming, and depending on file sizes possibly impractical.
- data security is compromised since corporate data resides outside of the realm of protection offered by MPE and IMAGE. This problem is compounded by the duplication factor mentioned above.

With the Batch Integrated approach to mini - micro integration we are still faced with the problem of protecting and securing the data which resides on the Personal Computer. The problem is aggravated with this second approach since we would be using this approach to tackle larger, more complex applications (and hence more critical data) than with the Standalone approach, and because there are potentially many copies of this unsecured data. We can partially offset these concerns with the consolation that the data left unsecured on a Personal Computer at any point in time is but a snapshot of the entire Database, and that the central IMAGE Database which contains the whole picture can be secured and protected.

The third approach we call OLRT (On-Line Real Time) Integrated. This solution is similar to the Batch Integrated solution with one very important difference. By introducing a transparent networking mechanism called Remote Dataset Capability, we can eliminate the need for data duplication, and do the mini to micro communications in real time.

Remote Dataset Capability is defined as: The ability within a Database to define a Dataset which is physically resident on a different CPU, and to access this Dataset in a fashion that is transparent to the application program.

This approach involves designing an application that will be distributed across any number of Personal Computers connected to an HP3000. As the application designer you choose where the processing will be done (all on the PC's, or shared between the PC's and the HP3000), and where the data will reside (all on the HP3000 or shared between the PC's and the HP3000).

Those Datasets that are to reside on the HP3000 in the central IMAGE Database are identified to the Personal Computer as a "Remote" Dataset. When the Personal Computer user is running the application, the micro and the mini communicate in a fashion that is transparent to the user. Any Database transactions involving the remote Dataset (reads, writes, deletes or updates) are passed to the HP3000 where the appropriate IMAGE intrinsic is executed and the results returned to the micro.

Using this arrangement, the Personal Computers are connected to the HP3000 via a terminal port. The HP3000 treats the port as an I/O device as opposed to a Job/Session device. A number of PC workstations can share the same port.

To illustrate with an example, consider an Order Processing application where the actual entry of customers' orders is to be distributed across a number of PC workstations. An IMAGE Database could be developed to support this application consisting mainly of the reference or master Datasets such as Customer and Product. The PC workstations would each have their own local Database, perhaps matching the structure of the IMAGE Database. The local Databases would define the Customer and Product Datasets as Remote Datasets. As orders are entered at the PC workstations, lookups and validations for Customers and Products are processed against the centralized copy of those files maintained in the IMAGE Database. The resultant order transaction records would be stored locally at each workstation. If desired, at any time the transactions could be uploaded from the PC workstations and consolidated within the IMAGE Database, where they would be available for centralized reporting.

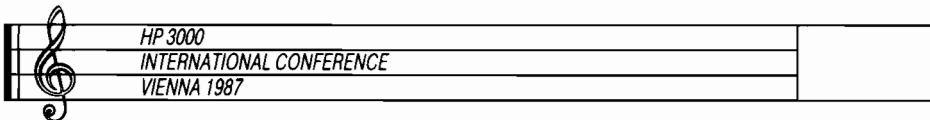
This On-Line Integration approach offers these benefits:

- distributed processing, shifting some of the processing load from the HP3000 to the PC's.
- information that is physically resident on the HP3000 can be read and updated immediately.
- there is no data duplication.
- the critical corporate data can be left on the HP3000 where it is protected by the security provisions of MPE and IMAGE. Access to the data is controlled by the application software.

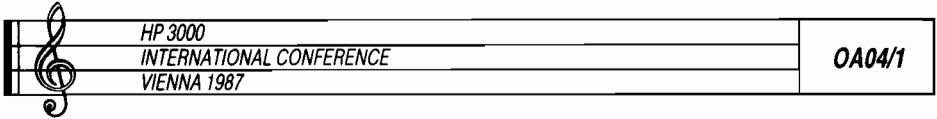
What is the appropriate mini - micro integration approach? All of the approaches offered in this paper support application development activities on either machine. Depending on the approach, the system processing will take place exclusively on the PC, exclusively on the HP3000, or will be shared between the two resources. Depending on the approach, the data will be physically resident on the PC, or the HP3000, or both and sometimes with duplication. Where the data resides can have serious impact on the security of the data. Obviously the choice of approach must be determined by your resource availability and your application needs. To add to the decision making complexity, consider also that the choices are not mutually exclusive. A mixture of approaches may be appropriate for the various components of a specific application.

Summary:

As this paper has pointed out, it is the case that we have at our disposal the technology to effectively integrate the use of our Personal Computers into our HP3000 based application system development and processing. In order to achieve this end of mini - micro integration we need to embrace the tools required: a 4GL programming language common to both the MPE and MS-DOS environments , a common DBMS, and transparent micro to mini communications. With these tools in place, we can design applications, distributing the processing and the data across a network of MPE and MS-DOS machines, with a tremendous amount of application design flexibility. The tools make it easy. The challenge is to maintain the high degree of data security that we have become accustomed to with the security provisions offered by MPE and IMAGE.



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987



PC to HP3000 Communications: A Perspective

Sam Patsy  
Hewlett - Packard

Having majored in the biological sciences, I had an in depth exposure to the subject of evolution. As a matter of fact, the book I am currently reading is Darwin's Origin of the Species, this time, for pleasure. This subject of evolution was brought home to me recently as I attended the SE training for the newest HP family member, the Spectrum series. For a prehistoric SE who remembers wiring your own board, paper tapes, and an increase from 4K to 8K of main memory as a heralded event, the capabilities of the Spectrum series are mind boggling.

We are all aware of the milestones in the evolution of our society. The change from an agricultural economy brought on by the Industrial Revolution was an unquestioned landmark. Since then we have witnessed several other changes in our society--political, economic, and social. In computer technology we have a slowly emerging species that is, perhaps, ready to become the Homo Sapiens of that industry--the PC.

In our technology of today, we speak of MIPS of CPU speed and gigabytes of peripheral storage. These and other advances in technology are making the dichotomy between the PC and the mainframe clearer. With these technological advances, cost effectiveness dictates that we let the big boys do what they do best and offload some functions to the PCs. If we looked at a list of applications running on the 3000, I am sure we could unanimously agree that some of these applications could be done more efficiently and effectively on a PC, thus allowing the 3000 to do what it does best and far better than a single or group of PCs could do--large data bases and communications to the outside world.

This is, however, a two-way street and the PC does offer some advantages that are indeed unique.

THE HP3000

In the past we logged on to the HP3000 from a terminal, did our thing and, if fortunate enough, had a slaved printer for our output; otherwise we treked to the computer room to pick up our output. If we were at home and logged on, we waited until morning at work to see our output. In this HP3000 world of one of many users, we are at the mercy of a concept called system performance, which is a sophisticated way of saying, how many



people are on the system and what are these people doing. When you type "RUN" on the HP3000, you no longer have control of your destiny. You are at the mercy of available memory, scheduling queues, I/O bandwidth and a host of other factors that range from unpleasant to unbearable. At this point you bootleg a copy of OPT or SOO and try to determine how to get more miles per gallon from your system.

#### THE PC

I had mentioned earlier the uniqueness of the PC. The attributes defining this uniqueness bear some discussion.

The most significant attribute of the PC is the fact that it operates in the "NOW". You do your work NOW, no performance issues, you are in control. The PC lets you be as good as you want to be. You can rerun your program as often as you like. Want to play "What-If" strategy on a problem? Go for it. I am convinced the name "Personal" was chosen for the PCs only because the "NOW" computer just doesn't have a good marketing ring to it. The philosopher would say that "NOWNESS" is the quintessence of the PC. The NOW concept reaches its peak in the portables. Last evening I watched as my son logged on to a mainframe at a local University; used Kermit to download his assignment to our HP150, and then began to write his Pascal program for his homework. The NOW concept is very, very real.

Another attribute of PCs is "user friendliness". Each software package released is easier to use than the previous one. Much of the competitiveness we see is based on "user friendliness". For many packages, the user friendliness makes the directions on a cake mix seem complex. The only prerequisite for using a PC is being able to read; with some of the graphics today, being able to read might not even be necessary.

Cost effectiveness must be considered as an attribute of the PC. It is difficult to be objective about cost. What I would pay for a bottle of wine may be significantly different from what you would pay for that same bottle. However, when we pay several hundred thousand dollars for a computer system, we would like to get the maximum utility from that system, i.e., let it do what it does best. Mainframe tasks that are done more efficiently elsewhere should be just that, done elsewhere. To the rescue comes the PC. Heavy resource users on the mainframe such as word processing, graphics, can be more efficiently and effectively done on the PC.

**BEST OF BOTH WORLDS**

Here we have two diverse environments, the PC and the HP3000, each outstanding in its own way. How can we merge the resources of these two environments to help our customer. Let's build a conceptual model of how a PC might talk to an HP3000. At first glance we have:

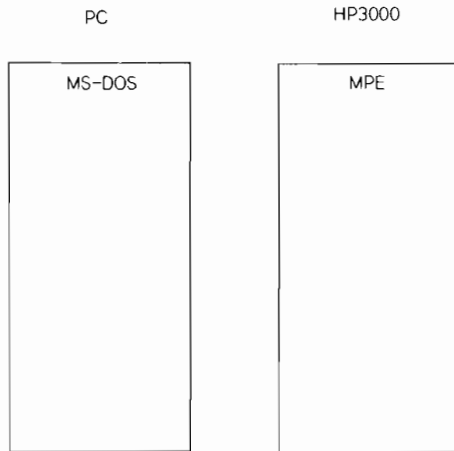


FIGURE 1

Rather quickly we recognize a problem of two different operating systems. We know from past experience we can use a PC as a terminal and log onto the HP3000. However, we want more than terminal capability; we want to have file transfer capability. We want to communicate with the various subsystems on the HP3000.

Our next logical step is:

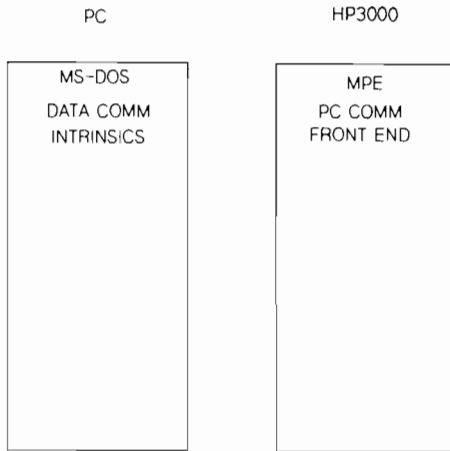


FIGURE 2

We have overcome our first hurdle; we can now allow the PC, as a PC, to talk to the HP3000. Now that we are on the HP3000, we want to access and share resources on the HP3000.

We then do the following:

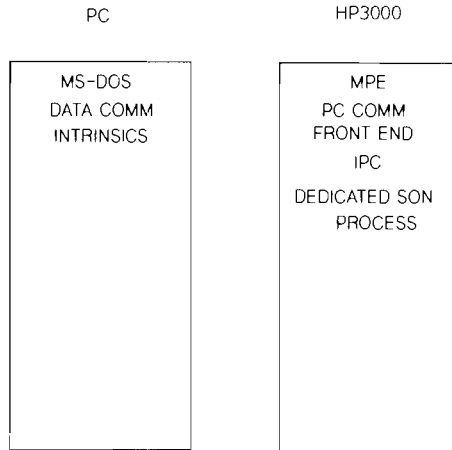


FIGURE 2

We have created a dedicated son process to talk to a specific subsystem. Perhaps we want to talk to the spooler, an Image data base, or a disc. The son process will use IPC files to talk to the PC communication Front End.

Our next step is:

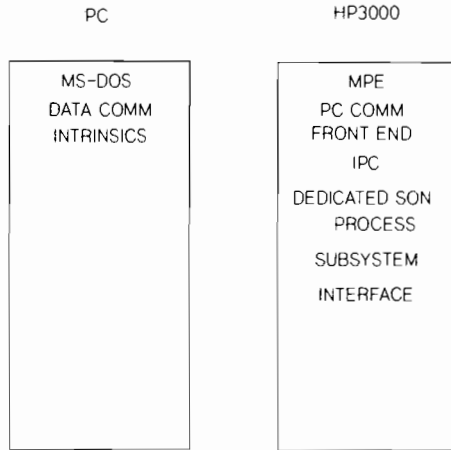


FIGURE 4

Our dedicated son process will then communicate with a specific subsystem interface.

Our final step is:

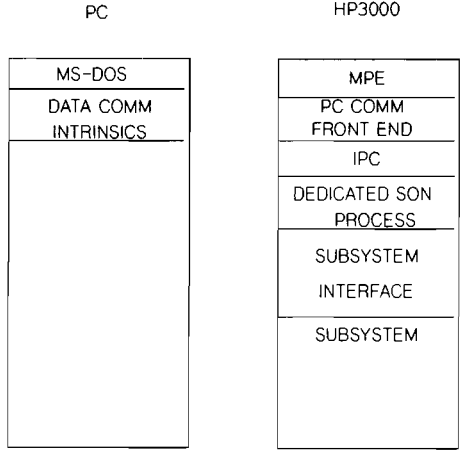



FIGURE 6

Viola!!! We now have a PC talking to a specific resource on the HP3000 and moving information between the PC and this system.

We now have the capabilities to share peripherals on the HP3000, as well as share the information on the HP3000. Again, this is a two-way street and we should not become stereotyped into thinking of the PC as some type of parasite leaching resources from the HP3000. Rather, this is a true symbiotic relationship; and becomes more so as we increase the capabilities of the PC.

	HP 3000	OA04/8
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

#### EXAMPLES OF PC-HP3000 CONNECTIVITY

The purpose of the PC-HP3000 connectivity is to take a resource, information, in this instance, and be able to move, distribute, query, and integrate it with other information. As mentioned, this is a two-way resource transfer. The information on the PC has as much intrinsic value as the information on the HP3000. Remember, this is a symbiotic relationship.


Some specific examples of this connectivity link may be helpful. Consider working at your PC and routing your output to a 2680 laser printer on an HP3000. Print Central allows the PC user to access any of the printer resources on the HP3000. From a menu driven screen, the PC user can toggle between a local printer and a printer on the HP3000. The same printer sharing capability is offered in Resource Sharing.

The next logical progression in peripheral sharing is to have the PC utilize the disc on the HP3000 as a resource. This capability is offered by Resource Sharing. A portion of the HP3000 disc space is reserved for the PC MS-DOS files. When a user logs on to the HP3000, his is either in the PC MS-DOS domain or in the MPE file domain. A utility is provided to convert between the different file formats.

Also, under Resource Sharing the PC can access the tape drives on the HP3000. This provides a convenient method for backing up the PC.

Another example of this connectivity is Information Access, which appropriately named, allows a PC user to extract information from an Image Data Base and bring that information over to the PC. The phrase, "bring that information over to the PC," is a conservative understatement. That phrase should be highlighted, underlined and set in flashing neon lights. In bringing "that information over to the PC", Information Access allows the PC user to choose one of several formats to have the data come to the PC. Think of that for a minute; extract selected data from an Image Data Base; bring that data over to your PC in, let's say, a Lotus format and immediately begin working on that data in a Lotus spreadsheet. If you will please, the flashing neon lights.

One of the more recent additions to this connectivity is Starlan. The Starlan products allow us to have several PCs networked to a PC server. This network can then be connected to an HP3000 Thicklan or HP3000 Thinlan. Once we have established a physical link to the HP3000, we can then, while on the PC network, establish a virtual terminal session on the HP3000. The virtual terminal session is an HP3000 user. We can now toggle amongst a standalone PC, a PC on a PC lan, and a PC as a virtual terminal on an HP3000 and not lose any of our connections. Excuse me, but could we borrow those flashing neon lights and better turn up the intensity--we have a big one here.

	HP 3000	OA04/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Everything we can do on a Starlan can be done on a Thinlan. Only the physical connection is different. Starlan uses twisted pair wiring, and Thinlan uses coax wiring.

So as not to forget the remote user, HP offers a product that allows a remote standalone PC user the capability to share resources on the HP3000. This is a virtual terminal session, by modem, offering all the horsepower of the HP3000 to an isolated user. This connectivity product is HP Serial Network.

Graphics connectivity is provided with the Curator product. Curator allows you to transfer graphics between a PC and the HP3000. Curator makes the necessary format changes such that when the conversion is completed and uploaded or downloaded, you have a graphic in a usable format on your respective system.


Similar to Curator is a utility that allows conversion of files between the PC domain on the HP3000 and the MPE domain of the HP3000 and vice versa.

So far we have concentrated on the PC-HP3000 world and the benefits of this link. Depending on your perspective, this could be the tip of the iceberg. Consider the communication capabilities of the HP3000 and the implication of these capabilities. Consider that the HP3000 can speak NRJE, IMF, MRJE, BISYNCH IMF, PROFS, DISOSS, and X.25. This essentially ties the HP3000 to the outside world. Similarly, this capability is passed on to the PCs connected to the HP3000.

Let's do an instant replay ...

We can be sitting at our PC connected to a server over a local area network; we can text a file from the server; we can merge this file with a file that is local to my PC; we can then convert graphics from the HP3000 and upload it to the PC; we can then merge the text and graphics; and, finally, send the output to a laser printer on the HP3000. Not too shabby!!!




	HP 3000	OA04/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

#### SUMMARY

The story of the PC can be best illustrated by an old cigarette advertisement--a two page ad. On the left-hand page was a black and white photo of some women in the 1920s, very dull and drab clothing of that era. On the right-hand page was a color photo of a voluptuous young lady, with a cigarette in her hand. The caption to the ad said, "you've come a long way baby". Well, for the PCs we can say, "you ain't seen nothin yet". Let me explain.

Hewlett-Packard does not issue crystal balls as standard equipment for its field SEs; however, let's look into the future. Immediately on the horizon is a 32-bit processor with a 16/32-bit data bus driving the PC. This processor will talk to a co-processor of similar architecture as well as a graphics co-processor. All of these processors will run at a modest 12-18 megahertz clock speed. Might as well throw in 8-16 megabytes of main memory. The fact that the dinosaurs could not adapt to change should be a forewarning to the MIS directors, managers, vice-presidents, etc. to recognize the capabilities of the PCs.

My eyes are gett-ing b-l-urr-ed, it is be-com-ing ver-y dif-fic-ult to wr-ite. I believe it is those flashing neon lights, they keep getting brighter and brighter and brigh-ter.

	HP 3000	OA05/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	


## DOCUMENT FILING AND RETRIEVAL

Rudi Huysmans  
Peter Arfeuille  
SYDES NV Mechelen  
Belgium

### Abstract

In the past, office automation has been characterized by the development of powerful word-processors, which often cause an exponential growth in the number of text files to be stored. During the last few years, more attention was paid to the filing and retrieval of these documents.

This lecture presents a system through which the user can not only store documents in a secured format, but can also define a meta-information structure by which the documents can be retrieved.

	HP 3000	OA05/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

When looking at the classical DP applications, we can recognize the same major functions :

- data entry : allows the user to capture information
- data management : allows the user to keep and to manage the information
- query processes : allow information retrieval

These DP - systems can handle information very well if it is completely structured. But as most information handled by office workers consists of both text and data, the classical solutions don't live up to the requirements.

The use of word-processors facilitated the production of text, resulting in an unmanageable collection of files with a growing redundancy and inconsistency, even leading to a parallel manual filing system.

The main goals of a document processor is to manage unstructured information in an efficient way, not only for users in an office environment, but for the DP department too.

## 2. Definition of the problem

To recapitulate the problems in office automation :

- an exponential growth in the number of files to be stored.
- inconsistency in the management of files, even leading to parallel manual filing systems
- difficult retrieval of the documents
- the impossibility to store additional information on the documents in an easy and user-friendly way
- a lot of lost disc space because of the presence of many blanks
- long back-up time because of the cutting up of files.

As you can see there is a great deal of work to be done in storing documents in a more compressed way and defining meta-information structures to retrieve the texts.

## 3. Nomenclature

### 3.1 document

A document is an unstructured file of all kinds (ASCII/binary, numbered/unnumbered, program object code, LOTUS 1-2-3, ...) that has to be stored and retrieved by a user.

### 3.2 library

A library is a set of documents. With each document, some additional information can be stored. This information follows a uniform pattern.

The pattern can be defined when creating the library. The user has the capability to change this pattern in an existing library.

### 3.3 attribute

An attribute is an element of the fixed information pattern of a library. A document can have one or several values for an attribute.

#### 4. The document management system.

##### 4.1. Definition of a document management system

In informatics, a database is a compilation of data, and the relationship between the latter.

Analogically to this definition, we now define a docubase.

This is a compilation of documents and data on these documents.

Comparing a docubase and IMAGE, we can make the following associations :

#### DOCUBASE

-----

documents  
attribute  
value of the attribute

#### IMAGE

-----

recordno  
data item  
value of the data item

First we summarise the necessary characteristics of the docubase.

1. The docubase must be able to keep documents. The identity of this documents should be any kind of MPE-file.

The possibility should exist to save documents in a compressed form. This means that a lot valuable disc space can be saved.

2. Appart from these documents, there should be a possibility to insert additional information under "attribute"-form with corresponding values.

Attributes should be allowed to consist of more than 1 value. Attributes can also have no value.

3. A querylanguage should be able to look up documents. This language should be substantially elaborated. Because of the great variety of criteria for looking up documents, the language should be as uniform as possible.

4. The structure has to be dynamic. It may not be frozen at creation time. (this is an important difference between the scheme of an image database and the scheme of a docubase).

5. The performance of a docubase should be acceptable.

6. There should be utilities like : recovery of the library, copy of libraries, print funtions, performance optimisation.

#### 4.2. Technical solutions

The simplest and the most used solution is to store every text in a separate file. The only way to retrieve a document is to know the name of that file. Document handling with this method is easy.

All you need is a very good memory and big directories for the endless list of files.

An improvement of this method is to maintain extra information on every file in a special file or in some database. You still have a sea of files on your computer.

The best solution is to have the text of all documents and all information about the documents in a single system. This eliminates the proliferation of files and the consistency problem of meta-information (i.e. information about documents).

One solution is to use an all-purpose database and to use the search mechanism of the database to retrieve your documents.

Although technically possible, this approach is likely to produce a big space overhead and poor performance because most classical databases were not designed for this kind of use.

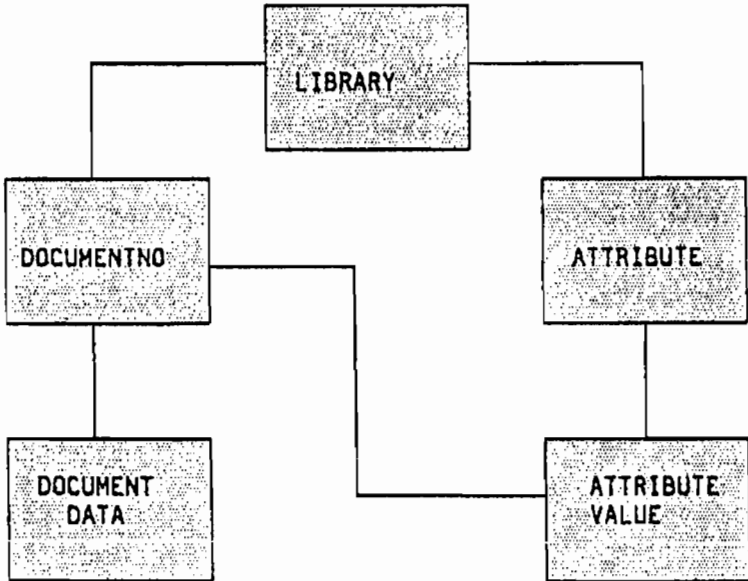
The best solution is to design a system especially to store documents and to maintain information about documents.



#### 4.3. The structure of a DOCUBASE

A DOCUBASE stores all documents and information about them in files called libraries.

Each library has the following conceptual structure:



- DOCUMENTNO** : Is a number identifying the document
- DOCUMENTDATA** : This is the 'text' of the document. This can be any type of MPE file with or without userlabels.  
(Not only textfiles, e.G. you can put a runnable program in a library, get it out again and run it).
- ATTRIBUTE** : Is a name describing a usefull information.  
Suppose you want a library to store memos.  
You could define the attributes FROM, TO, DATE, SUBJECT.
- ATTRIBUTEVALUE** : For a certain document and attribute you can add a value.  
example: we add the following values
- | document | attribute | value |
|----------|-----------|-------|
| 0        | FROM      | JIM   |
| 1        | FROM      | JIM   |
| 1        | TO        | MARY  |
| 2        | FROM      | JOHN  |
| 2        | TO        | MARY  |



An attribute together with its values is in fact a table of the following form:

FROM:

attribute value	0	1	2	3	4	.....	max documentno.
.....	-----						
JIM	1	1	0	0	0		0
JOHN	0	0	1	0	0		0
.....	-----						

A 1 means: the corresponding document has this value for the attribute.

A row in the table tells you what documents have the value at the left, a column tells you which values a document has.

In the library itself this table is stored in a compressed form. The method used to compress these tables makes it very efficient to read a row but somewhat less to read a column.

This choice was made deliberately because the main purpose of the attribute structure is to retrieve documents with certain associated values.

The number of values per attribute is only limited by the maximum size of the library file.

As you can see the main idea is conceptually very elegant and simple, yet very general in its possible applications and open-ended because the user is free to define any attribute meaningful for his application. He is not restricted to predefined attributes associated with his documents. Notice also the fact that every attribute has the same query capability. These are no privileged attributes.



## 5. An example

To clarify the concepts mentioned before, here is an example :

Suppose we want to maintain our correspondence.

A document is a text file created and manipulated by use of an editor or a word processor.


### 5.1 First the docubase is designed.

We can define e.g. four attributes

- AUTHOR : single-value attribute  
length of value : 16 characters  
= the name of the author of the text
- CUSTOMER : single-value attribute  
length of value : 16 characters  
= name of the addressee of the text
- DATE : single-value attribute  
length of value : 6 characters  
= creation date of the text
- SUBJECT : multi-value attribute  
length of value : 16 characters  
= some keywords to give an idea of the contents of the text

The difference between a single-value attribute and a multi-value attribute is that a document can have several values for a multi-value attribute but only one value for a single-value attribute.

However, that doesn't mean that a single-value attribute is a key in the sense of IMAGE and KSAM, because several documents can have the same value for the attribute AUTHOR.

	HP 3000	OA05/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

5.2 After having created the library we can start adding documents and give them values for the attributes.

DOC 0    AUTHOR    :    HOLMES  
          CUSTOMER    :    HP  
          DATE        :    860224  
          SUBJECT    :    SPECTRUM  
                          COMPUTER  
                          HARDWARE

DOC 1    AUTHOR    :    CAMPBELL  
          CUSTOMER    :    PENTECH  
          DATE        :    860316  
          SUBJECT    :    VECTRA  
                          discount  
                          installation date

DOC 2    AUTHOR    :    ROGERS  
          CUSTOMER    :    GOVERNMENT  
          DATE        :    860321  
          SUBJECT    :    SALES  
                          BALANCE

DOC 3    AUTHOR    :    ROGERS  
          DATE        :    860401  
          SUBJECT    :    LOANS

Note that document 3 has no value for CUSTOMER. This means we can have documents without a value for attributes that are meaningless to the documents. It is up to an application to allow or disallow this.

### 5.3 Retrieval of documents

The power of the Query language and the performance of the Query execution were two of the main objectives in the design of a docubase retrieval mechanism.

It should be possible to ask questions like :

- Which documents have 'Rogers' as the AUTHOR ?
- Which documents have 'balance' as a SUBJECT ?
- Which documents have 'Rogers' as the AUTHOR and have been written between 860101 and 860601 ?
- Which documents have 'loans' or 'balance' as a SUBJECT and are not written by 'Holmes' ?

### 5.4 Possible actions

We can delete a document. Watch out ! Your beautiful text will have disappeared completely.

We can text our document with our favorite editor or word-processor, to make it even more beautiful.

We can delete, add or change the values of the attributes.

### 5.5 Management of the docubase

Suppose that at a certain moment we are not completely satisfied with the docubase anymore.

Then we can add attributes to widen the range of queries. The existing documents will have no value for the new attributes. The docubase has a number of possibilities to copy the values of one attribute to another, or to initialise the values.

If we delete an attribute all the values for this attribute will disappear to.



## 6. The docubase programmer interface

To use the docubase, the programmer needs a comprehensive set of commands : the docubase intrinsics.

Possible commands are :

### Library commands

openlib/createlib  
deletelib  
closelib  
infolib

setbits

### Attribute commands

bitmaps

addattribute  
deleteattribute  
copyattribute  
renameattribute  
initattribute  
infoattribute

ributes

### Document commands

ributes

create/open  
deletedoc  
getdoc  
putdoc  
closedoc  
getdoc-to-file  
putdoc-from-file  
lockdoc  
unlockdoc  
checklock

### Value commands

addvalue  
deletevalue  
getvalue  
updatevalue

### Query commands

docquery

## 7. A comparison between a docubase and IMAGE

A very important question to ask is why and when to use a docubase and when not.

All computer use has to do with the manipulation of information. We can define an information system as a system to store information, to retrieve it and to change the information in the course of time. This is a rather general description which fits a lot of systems.

A good old MPE file is an information system. It stores records and with the proper intrinsics we can retrieve the record and update it. An Image database is a more elaborate information system. Basically it stores records (consisting of several items) but lets you retrieve them in a much more sophisticated way (dbfind, dbget, etc.) and update them (dbput, dbupdate, etc.)

A docubase stores documents, it can create, delete, store, update and retrieve them. With the help of attributes and values it can find them using a wide variety of the docubase queries.

All information systems have their typical merits and weaknesses. An important consideration is what type of query we can do and how fast the document will be found. Most information systems have a number of built-in search mechanisms that are fast but only work for certain types of queries. More complicated queries are done by combining several elementary operations and sometimes this can be very costly in performance.

For instance, searching a record with a particular record number in an MPE file is a fast operation but finding a record with a certain content can be very costly because we have to scan the file sequentially.

To compare a docubase and IMAGE we make the following associations :

Docubase	IMAGE
-----	-----
documentno.	recordno.
attribute	dataitem
value of the attribute	value of the dataitem

For a multi-value attribute we can have an unknown number of values in a docubase whereas in IMAGE we don't have an equivalent possibility. For the sake of simplicity let us ignore this for the moment. With the comparison in mind, we consider the following query : 'Which documents have 'ROGERS' as the AUTHOR ?'

- with IMAGE

with AUTHOR as a key we can find a chain for 'ROGERS' and by chained read pick up all the document numbers.

In IMAGE a data item must be a search item to have a fast access. A 'normal' data item can only be found with a sequential search.

- with a docubase


The docubase searches for the value and immediately has an answer for all the documents in the form of a (compressed) bitmap.

This is done on a uniform basis for all values and for more complicated queries that are much more difficult in IMAGE, where certain combinations and iterations of the IMAGE intrinsics will be necessary.

Another key factor in the design of docubase is the capability to add and delete attributes in an existing library whereas in IMAGE applications the structure has been frozen at creation time. Souplesse in the structure can only be achieved by application dependant code.

Where lies the strength of a docubase ?

- in the ability to manipulate the documents without creating an unmanageable collection of separate MPE files.
- in the richness of its query language and the performance of the queries
- in the possibility to change the structure of the attribute mixture during the lifetime of the library.
- a backup of a library in nothing more than a STORE and RESTORE of a single MPE file
- in the ability to keep files in a compressed way.

	HP 3000	OA05/15
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Biography

Rudi Huysmans was born in 1953 in Mortsel near Antwerp. After High School he studied civil engineer in Computer Sciences at the Catholic University of Leuven. He joined SYDES in 1982 as a commercial project leader. He is Vice Chairman of the Belgian National Users Group. Off business hours he works as a teacher in the Computer Sciences.

Peter Arfeuille was born in 1955 in Tielt. After High School he studied mathematics at our own "Antwerp University". He joined SYDES in 1984 and is working there as system analyst. His principal interests are application of theoretical know-how in real-life problems.








HP 3000

INTERNATIONAL CONFERENCE

VIENNA 1987

	HP 3000	<b>OA06/1</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Sophisticated CAP (Computer Aided Publishing) in a Commercial Environment

Dipl.-Ing. Dr. Wolfgang Vitovec  
 Telefon- und Adreßbuchverlag „HEROLD“  
 Wipplingerstraße 14  
 A-1010 Vienna  
 Austria

### Preface

I am always surprised how various publications make such flippant use of the terms "Electronic Publishing" (EP), "Desktop Publishing" (DTP) and "Computer Aided Publishing" (CAP). Indeed, every time I come across an article on the subject of "Electronic Publishing" I begin reading avidly, eager to learn what the author has achieved, only - but invariably - to find that his "Electronic Publishing System" boils down to nothing more than the production of a printed page. However, since to me the term "publishing" means more than simply the preparation of a document, I also expect "electronic publishing" to mean something more - namely an electronic distribution function to whatever medium concerned (be it via databases, BTX (videotex) or on an other medium in a machine-readable form). Whilst nowadays the term "publishing" is usually associated with paper as the medium, one of the most successful publishers of all times used a quite different medium - namely Moses.

Rather than the terms EP, DTP and CAP I prefer to use the term "Document Preparation" which also includes the (computer) typesetting - and this particularly so since DTP and typesetting very often differ only in the output medium and the output quality. However, it seems scarcely likely that my preference here has much chance of gaining wide acceptance, particularly since the abbreviation "DP" (= Data Processing) is already very firmly established.

This present work is concerned with the preparation of documents in typesetting quality for subsequent reproduction by printing - in other words, with the production of copies whereby the integration of graphics plays an important rôle.

### Terminology

In the context of this present work the term "graphic" should be understood to mean only "graphics of real objects" whereby these include, for example, signets, logos, fancy type and drawings. Where "graphics of synthetic objects" are meant, I have used the term "computer graphics" (Note: the distinction between graphics of real objects and those of synthetic objects - as well as the terms themselves - was adopted by Foley [FOLE84] - whereby Foley takes computer graphics to mean abstract quantities of lines, curves and planes). Figure 1 features a selection of "real object graphics".




*Fig. 1 A selection of "real object graphics"*

The description "image" will only be used here in connexion with continuous-tone presentations - photographs, for example (Note: within the scope of this present work the processing of images differ only in the necessarily extended resolution for that of graphics, so the description "graphic" is frequently used also to mean image!). Where the term "text" is used here it should be understood in the narrower sense, in other words, rather in the sense of "typesetting" and not in the broader telematic sense which includes also graphic, image and tone. This corresponds to the terminology used in typography and reproduction technology. Important also is the difference between book work i.e. the typesetting of novels, scientific works, ect., - and job printing - generally understood to be the printing of forms, tables, posters, ads etc.

### Introduction

In my own case the problem is the need to deliver copies comprising graphic, image, and text for telephone directories within the shortest possible time. This problem can only be solved given fully automatic preparation of entire, ready for the press pages i.e. through automatic make up and montage of the text including the graphics and images featured therein.

These past few years I have worked on the automation of typesetting and developed a book-oriented typesetting system for a commercial computer system (Hewlett-Packard HP3000). The typesetting program I developed - and which is subsequently referred to as either make up program or formatter - automatically creates lines and columns, automatically makes these up into multi-column pages, and automatically generates the page numbers, as well as two-level running titles and column headings dependent on text content - the positioning of these also being fully automatic.

	HP 3000	<b>0A06/3</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

A novelty in this system is the philosophy of can-split-positions, incorporated in an "endless" column whereby the "can-split-positions" are either explicitly defined or are automatically generated by the system. This approach enables a high degree of automation to be achieved with the make up - also in the case of mixed matter and job printing. At present the percentage of pages which require subsequent correction because they are not in keeping with aesthetic requirements is in the region of 0.5 %! Thus all these automatic functions are performed whilst observing the highest possible aesthetic standards.

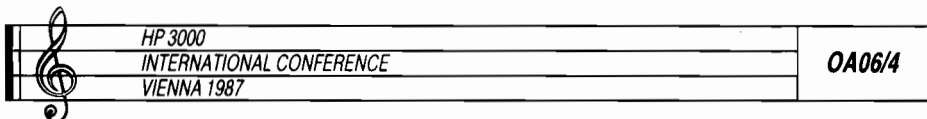
Since the implementation of the first version of this system roughly two and a half years ago, this system has created approximately 25.000 A4 pages for the telephone directory production. This present work was also produced using the same system.

When recording or processing the text, the compositor will leave vertical and/or horizontal blank spaces where graphics or images occur in the text. The compositor will determine the dimensions of the required blank space either directly from the manuscript or will obtain the necessary information (manually) from the graphic design department. At the make up stage the program will automatically leave a correctly sized space at the correct place for the graphics (signets, logos, fancy types, etc..) and images (autotypies) which will subsequently be pasted into position in the exposed page by the montage department. In order to avoid this last manually performed montage job, it is necessary to have graphics and images digitized and integrated into the make up and exposure process.

There is a number of electronic page montage systems on today's market. The underlying principle of these systems is the digitizing of all informations to be processed for the production of a page [SPRI82]. This means that the texts also have to be scanned (and stored as bitmaps). The storage capacity taken up by these "text bitmaps" is approximately 100 KB for a text page, and for a magazine page with black/white illustrations approximately 500 KB. The storage requirement is so big because a character requires between 20 - 50 bytes after scanning. An A4-sized character-coded page of text (i.e. with 1 byte per character) requires on average 2 - 5 KB.

By comparison, for the typesetting of a telephone directory page comprising only text in character-coded form, only 10 - 20 KB are necessary, whereby the text commands are included. However, since the storage requirement influences not only the disc capacity but also the processing time, I consider it more expedient to leave the text as a character file - i.e. not to scan the text but rather to leave it character-coded - and to incorporate the graphics and continuous tones (as halftones) therein.

This present work describes a method whereby graphics and images are stored in a database, automatically incorporated into the text and subsequently taken into consideration at the make up stage with the result that complete graphic-image-text pages are obtained. Consequently there is no need for the compositor to create



blank spaces manually for the graphics and images, and the time-consuming pasting in of graphics and images in the exposed page is also avoided.

### **The Model**

In the case of data involved in the production of telephone directories we can distinguish between editorial data and order-related data. Examples of editorial data (non-composition) include town data with name, postal code, county, dialling code, etc., - classified headings (column headings), but also the "normal" (i.e. unpaid) subscriber entries. Since these same data will be required for several different directories and sections of directories - e.g. on the one hand for regional and supra-regional directories, and on the other hand for various directory sections such as the local and classified pages, the data in question are managed on a composition-independent basis i.e. they are stored in the database without considering the final layout of the directories (Note; the various directories and directory sections differ from one another not only in the column width but also in the type and in the editing of the data!).

By contrast, the order-related data (i.e. the ads - "typesetting") which are individually designed in accordance with the customer's wishes for printing in a specific directory, and thus must be managed with text (formatter) commands for a specific layout whereby after delivery of the galley proof alterations will normally only be made at the customer's request. Such an ad would logically appear on one or several different pages in one or several different directories. In other words, it will be determined in advance in which directory, in which directory section, in which branch (column), in which town, under which dialling code, and under which name the advertisement should appear and whether it should replace the corresponding editorial subscriber entry.

**Graphics and images** can be included in the editorial data - for example in the form of city insignia or panoramas (as a part of town-related texts) - or they may occur in ads as logos, signets, or fancy types, or even as continuous tones (e.g. photographs). The more voluminous graphic or image data are thus (in keeping with the frequency of the ad concerned) likely to be needed once or several times. With the object of economizing on storage, it is necessary to store these graphics and images in an own database (where they are stored in digitized form after having been digitized by a scanner or digital camera system and provided with an identification key) independent of the editorial data. With the help of a text command it is then possible to recall a given graphic and incorporate it in a given text, whereby the ultimate positioning of the graphic will depend on the (graphic incorporating) text command.

Subsequently when producing copies for a particular directory the editorial data (prepared for the specific directory in question) and order-related data are brought together with the graphic data. However, with the object of keeping the data volume at the make up stage as small as possible, only those graphic data necessary for the make up - essentially the height and width of the smallest rectangle able to encompass the graphic - are incorporated into the other data. The effective (physical) integration of the graphics and images does not actually occur until the production on the typesetter or printer of the made up (and converted) pages.



The following figure shows how the various components are kept separate during processing and then "merged" for the output. The functions "DC" and "Telecopy" are features of future system extensions.

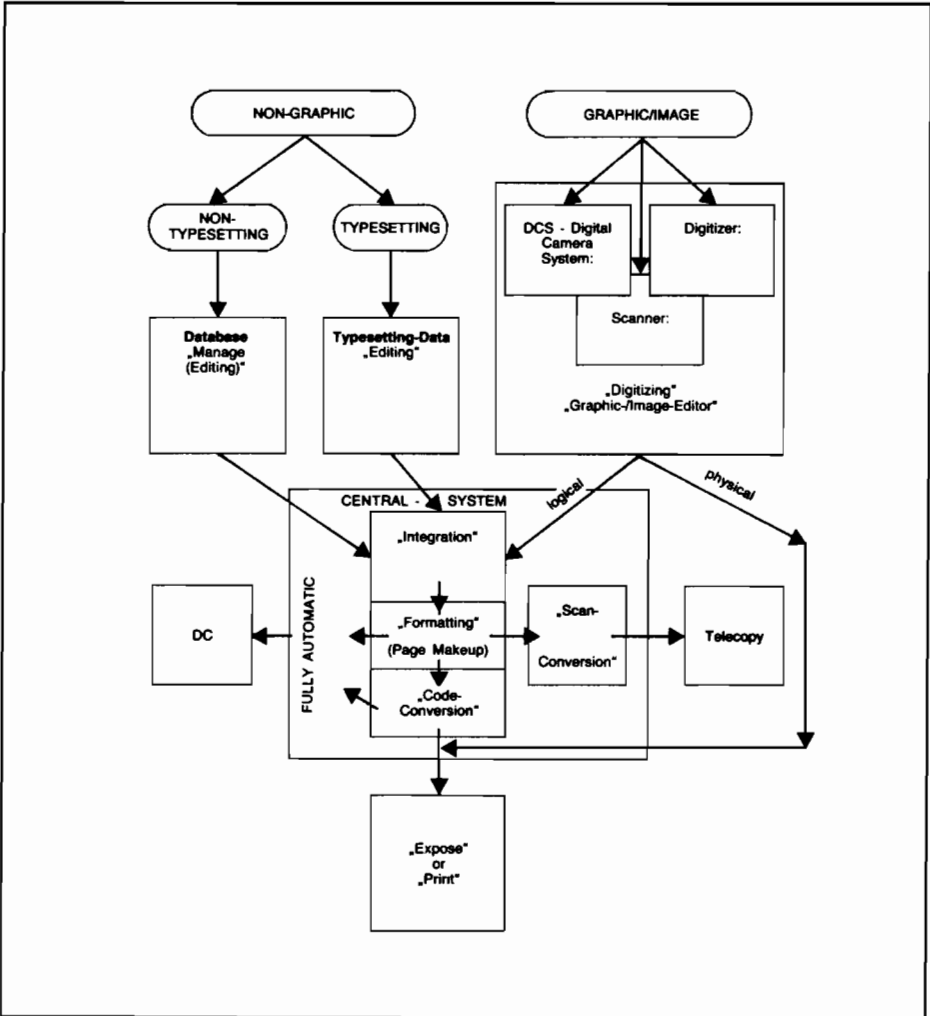


Fig. 2 The components of an integrated Graphic-Image-Typesetting-System




This model entails a major problem in that a system of this nature can only function efficiently when it is possible to find suitable presentation forms for text, image and graphic at the individual processing stages. If, for example, the run-length codings for the storage and transmission of graphic and image are optimal - because they minimize storage requirements - then the byte/pixel coding will be found more suitable for the editing (marginal corrections, etc.) of graphics [FOLE84]. For formatting purposes, on the other hand, it is sufficient to know the horizontal and vertical dimensions of the image and graphic - whereby it is usually sufficient to describe the image and graphic by means of the smallest rectangle which would encompass them. The compositor, who has to set the type for a given graphic, will not normally need to know the content of the given graphic, but he should at least be given the enclosing rectangle dimensions. Generally speaking it is not necessary - and also not expedient - to provide the compositor with the actual graphic or image (sized 1:1) since for him the nature of the image is only of secondary importance. Relevant here is only the logical text object "image" which takes up a specific space - i.e. will be projected (mapped) on a specific topological structure.

Much the same applies for the presentation of the set text: for the compositor operating with computer-aided book typesetting (i.e. the setting of straight text) at the most the setting of discrete hyphenation marks and thus exerting an influence on the hyphenation is of importance - meaning in effect that he need not concern himself with the 1:x presentation of the ultimate appearance of the text - which can only be achieved with very considerable effort. Instead he is only concerned with the information as to how the words should be hyphenated. In the case of job printing, on the other hand, it is important for the compositor to see whether one or more given words e.g. a title in special wide or big letters, can in fact be accommodated in a single line as intended. At least he must see on the terminal whether or not it will fit in one line. On the other hand, he will manage the composition of formulas more easily when he has an 1:x presentation of the set text in front of him (Appelt describes in [APPE85] a text system which permits the simultaneous editing and presentation of a made up text). However, for the graphic designer, whose job it is to develop a signet for a given text, the content matter of the text itself is of secondary importance only.

The following pages feature a more detailed description of how graphic data are stored and the nature of the make up model serving as the basis for the formatter which allows the automatic making up of data having different structures.

### **The scanning and coding of graphics**

At this point the author would like to take the opportunity to express his gratitude to Messrs. Agfa-Gevaert who kindly made the "Agfa S200pc Image Scanner" available for the purpose of the described tests. This handy and good-value-for-money (priced at approx. 100.000 Austrian Schillings) black/white flat-bed scanner permits the digitizing of copies to a size of 210 x 350 mm (approx. 8.4 x 13.7 inches) at three different resolutions (half-standard, standard and double-standard) whereby the standard resolution can be adjusted by the manufacturer to 150, 200, 203, or 240 dots/inch. The scanner was connected to an HP150 via the two RS-232-C serial ports - whereby one connection serves for the transmission of the scanned data to the

	HP 3000	<b>OA06/7</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

HP150 with 9600 baud (here a parallel port with a transfer speed of 500 KByte/sec is also possible); the second connection is for the purpose of scanner control. Once the original has been inserted the scanner can be controlled entirely via the HP150 keyboard (and/or Touchscreen) with the help of a suitable software.

In addition to a rough resolving preview function there are further facilities for window selection, as well as for the selection of intensity, contrast and coding. The scanner delivers the digitized graphics either in the form of a bit map or as run-lengths (Note: "run" is understood to be a sequence of equally light (same shade) image elements (pixels) of an image line; Run-length (RL) is the term used to describe the number of pixels in a run). There are different one and two-dimensional run-length codings which - particularly for purposes of datacommunication - are used to reduce the redundancy. With the help of a demo-program from Agfa-Gevaert the scanned graphics are stored on disc and later transferred to the HP3000 - as binary files - with HP-AdvanceLink. Once on the HP3000, the RL-coded graphics are loaded into an IMAGE database (see the following chapter). The scanner gives good results across the black/white range (perfectly satisfactory for office automation purposes). For continuous tones - because of the too-low resolution - it is certainly only good enough for the office automation sector.

Yet another possibility for the digitizing of graphics is provided by the "Digital Camera System" of HP with a resolution of approximately 300 dots/inch. An application is described by Abelow in [ABLE86]. The data compression used in this case is a variation of the Quadtree coding [PURG85]. Here a graphic is divided into rectangles whereby each rectangle containing (black) dots of the image is again divided into further smaller rectangles.

Whereas the data volume with bitmap coding increases quadratically with the resolution (respectively with the size of the graphic), the increase in the case of RL- and Quadtree-coding is rather linear. There are also outline-codes where the data volume is not related to the resolution or size. A format very often used for storing digitized types is the Ikarus format [GARM86]. This format facilitates particularly a whole range of transformations such as rotation and scaling. As a general rule it can be said that the greater the data compression is, the greater will be the effort necessary to convert the coding supplied by the scanner. The RL codes have shown themselves to be practical for the storing of graphics in an IMAGE database.

### **Management of graphic data**

If the scanner (or Digital Camera System) used to digitize the graphics does not supply data in run-length-coded form or in the required run-length coding, the data must first be suitably prepared. The nature and method of this preparation depend to a very large extent upon the used hardware. The digitized and suitably prepared graphics are then stored in a database.

In the case of the system described here, an 8-bit run-length code is used. Every digitized graphic is supplemented with an identification key and filed in the database under this whereby each graphic is stored within one master entry and one or more run-length entries.



The master entry of a graphic comprises the following data:

- .) the graphic identification number (as unique key)
- .) an alphabetical key (e.g. customer's name for a signet)
- .) the resolution (in form of a key)
- .) the number of (run-length) lines
- .) the number of columns (= number of pixels per run-length line)
- .) a wordmark which defines whether it is a positive or negative presentation
- .) a wordmark for a background pattern (?)

From the number of lines and columns, and from the resolution it is possible to determine the height and width, which are important for the make up.

The run-length data records contain the actual image information, or as the name suggests, the run-lengths. A number of run-lengths are stored per data record. Theoretically the required storage space will be at its lowest when the number of run-lengths per data record is equal to the average number of run-lengths per graphic.

When determining the run-lengths it is assumed that every line starts with a white run (provided that it is a positive presentation). However, this arrangement is quite an arbitrary one. We could just as well assume that every line begins with a black run. The black and white runs in a given line take over from each other per definition.

Now, should a line start with a black run, a white run of zero length must be inserted in front of the black run. The individual lines are always terminated by a return code - which explains why the horizontal width is also required in the case of negative presentations. The run-lengths are stored sequentially in the database records, and so one database record may contain a part of a "graphic line" or several lines. If the positive/negative wordmark in the master entry is set at "negative", for the purpose of further data processing it will be assumed that every line starts with a black run.


### **Management of image data**

The management of image data is largely analogous to that already described for graphic data. The only real difference is that the colour of the runs is not predefined - since white and black do not alternate here but instead the runs may be of any tone value. This means that the relevant tone value for a given run must be stored in addition to the length of that run.

Generally speaking an 8 bit codeword will be found sufficient for the tone value. This offers a fineness of 256 tone steps whereby 0 = white through to 256 = black.

### **Page makeup**

The structure and functioning of the makeup system (Note: in a word processing context, systems which make up text (or format them) are described as "formatters" or "formatting systems") constitute a key aspect

	HP 3000	<b>OA06/9</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

of this present integrated system. The basic idea of the formatter in the system under discussion here is that the text to be made up comprises a number of blocks whereby these blocks are made up of text parts which may not be positioned in different columns or on different pages. Blocks comprising text elements which may be positioned in different columns - i.e. the text part starts in one column but may end in another - are an exception and will be discussed separately.

Essential objectives of the make-up are:

- .) to ensure that the length of made up columns corresponds with the defined column length;
- .) to keep the number of pages as low as possible - meaning here that the makeup be as space-saving as possible;

Also, in the case of alphabetical reference works there is a necessary (but somewhat restricting) secondary condition:

- .) that the alphabetical sequence be strictly observed!

The makeup forms described in detail on the following pages, namely, "block makeup", "skirting board makeup", and "widow-free makeup" are seldom encountered in their pure forms but more often in mixed form.

### **Block makeup**

Whereas in newspaper or job printing one starts out with a finite data volume, in the case of bookwork the data volume by comparison is quasi "endless" - the directory may extend to three, 80, 1000, or even more pages.


For the multi-column bookwork page makeup the initial assumption made was that there is an "endless" column in which text objects are contained which must not become altered in layout during the page makeup process - examples of such text objects would be job printing parts, graphics and images. This means that in the endless column there are text objects which must not be split into parts i.e. it would not be permissible to have parts of the object appearing in different columns or on different pages. (Pure "text" blocks which must not or should not be split up could be, for example, text parts which together form a logical unit - such as lists, multi-line forms, or an entire multi-line telephone directory entry comprising name, address, and telephone number).

This problem can be formulated otherwise in that one might say that in the "endless" column there are "can-split-positions" embedded at which a new column or page can be started.

The text objects between the "can-split-positions" will be called "blocks" here. In order to achieve the desired column height, the formatter inserts "wedges" of suitable size at the "can-split-positions". These wedges can be compared with the "glue" and the blocks with the "boxes" of Knuth [KNUT79 and KNUT84].

### **Calculating the block-length**

The block length is determined from the "width calculation" and explicit text (formatter) commands which result in a vertical movement (in positive or negative direction). The "width calculation" is understood to mean

	HP 3000	OA06/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

the exact determination of the word or line length and the resulting line formatting. Thus the line arrangement is derived from the width calculation.

When calculating the block length, graphics and images are treated as text commands which cause a vertical movement in the positive direction corresponding to the height of the graphic or image.

### Column "white out"

In block page makeup the blocks (from the "endless" column) are arranged one under the other as long as the length of the new column is not greater than the defined column length. This usually results in there being some unused space at the foot of the column which will mean that the newly made up column does not satisfy the required column height. The procedure adopted in order to achieve the required column height is known as "white out". Per definitionem, white out may only be resorted to at "can-split-positions", or in other words, between blocks.

To start with, each block is allocated an immediately preceding "wedge" (i.e. space of variable size in this case in the vertical direction). The size of this wedge will amount to zero if the block is the first block in a column, and will be of only minimum value in normal cases. As and where required the formatter will increase the size of the wedges so that the column height is finally achieved (whereby the prescribed maxima for the wedges must not be exceeded). By definition of different wedge types - and thus also different block types - a hierarchical disposition of the blocks - with different minimum and maximum wedge values is achieved (relative to the layout). If, for example, for the different levels of a logical hierarchy - such as chapter / paragraph / break - different wedges and block types are used, and for the different wedge types are allocated different minima and maxima values, this will automatically lead to a hierarchy in the layout: the distances before a chapter beginning will always be greater than those preceding a new paragraph which in turn will be greater than the breaks.

Knuth [KNUT79] argued from the assumption that there is an optimal value  $K_{opt}$  as well as a minimum value  $K_{min}$  and a maximum value  $K_{max}$ . Taking into account the economic factor to the effect that "the fewer the pages, the lower the production costs", it follows that  $K_{opt}$  must be equal to  $K_{min}$ . Appropriate values for the minimum and maximum wedges of the various types for different types of work can only be determined on an empirical basis.

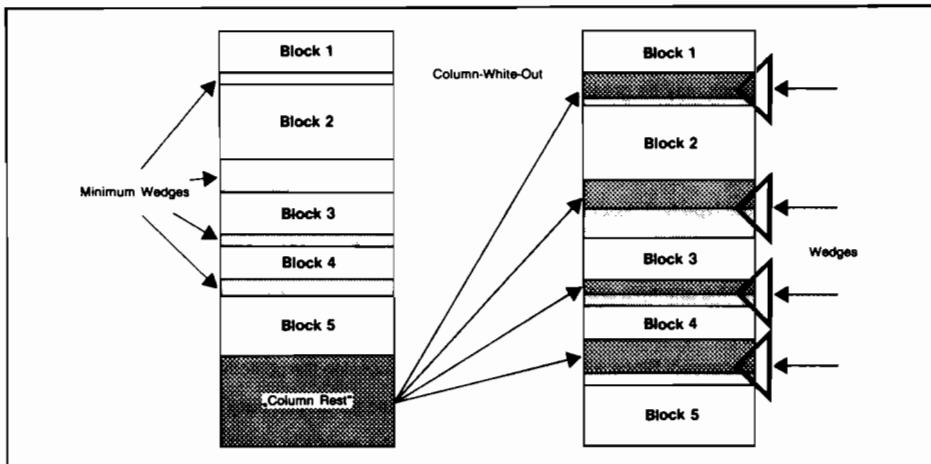


Fig. 3 Column white out

For the allocation of the column rest to the individual wedges there are various algorithms to which we may resort. In the currently used version the system is to white out the wedges highest in the hierarchy to a maximum - although it would be equally feasible to carry out a proportional white out (which would result in a completely different final appearance).

#### The output of the columns

Once whitening out has been effected the column is released in the output file. This is also the last opportunity to check out the vertical values - which only become known subsequent to the white-out (since the blocks are moved out of position during white out). The output of the columns is initially provisional until the page is closed. In some cases the data are erased again (e.g. in the case of skirting board page makeup).

If the EOF mark is reached in the output file, then the old file will automatically be closed and a new output file opened - whereby up to 999 different auxiliary files can be created. The size of the output file is generally so selected that it can accommodate ten A4 telephone directory pages (Always after ten pages the output file is automatically closed and thus immediately accessible for further processing while the makeup is still in progress!). Thus it is possible to make up 10.000 pages of telephone directory without intervention. Should it happen that the auxiliary data file are renamed in the meantime, they can be used by the system again and the makeup can then be continued for any given number of pages. Should the system find that no further auxiliary files are free, or that there is not enough disc space available, the makeup will be interrupted and an appropriate report passed on automatically to the operator. The operator gives the system the OK to restart processing once the conflict situation has been resolved. (Note: should for some reason the makeup be truncated, there is always the possibility to carry on with the makeup by simulating already made up pages (this is particularly time-saving where there are a great number of pages.)

### **Conflict situations during the block-makeup**

Initially two conflict situations arise with the above described algorithm:

- 1) the length of a block may be greater than the defined column length;
- 2) the sum of all maximum wedges (of a column for whiteout) amounts to less than the column residual space;

Conflict situations of the type first described are not readily solved in the case of pure block page makeup. However, in certain cases there is a solution possible (see widow-free page makeup).

In various makeup systems attempts are made to resolve conflict situations similar to the 2nd type above by making up anew, with revised values (e.g. for the space between words) in order to gain space for the column currently being processed. Since with the system under discussion the objective (see above) is to make up "saving as much space as possible" no additional space could be clawed back for the column currently being processed by a re-makeup of already made up pages. This means in effect that if a conflict situation of the 2nd type should arise, and there is free space at the foot of the column this would then be filled out with an own advertisement (or other filling material) manually in the montage department (this procedure will also be automated).

Generally speaking, in the case of block page makeup a possible problem situation could arise when the sum of the block lengths of two consecutive blocks turns out to be greater than the defined column length. Again generally speaking, block page makeup leads to an optically optimal result when the block lengths are relatively short in porportion to the defined column length.


### **Skirting board page makeup**

The skirting boards are a special case whereby we find blocks whose width is equal to the page width and which are positioned at the end of a page (as for example footnotes). Where skirting boards appear, the defined column length is reduced by the height of the skirting board. Should it happen that one or more columns on a page have been already made up and (provisionally) output, these must now be erased and made up anew. Should it turn out that a skirting board cannot be placed on the current page (e.g. the blocks on this page are too long) then the skirting board is moved to the next page.

At the moment a maximum of ten skirting boards per page can be processed. A special case is seen in that of skirting boards of a height equal to the defined page height. As a general rule it can be said that the conflict situations of the block page makeup occur to an even greater extent with skirting board page makeup (because of the reduced column length).

### **Widow-free page makeup**

In an effort to reduce the conflict situations arising during the page makeup of long text blocks comprising "straight text" (i.e. pure book work - as in the case of novels, etc.) a possibility to interrupt the block page makeup was created. If the block page makeup mode is switched off, the makeup system will automatically generate a can-split-position at the end of each line. This can lead to the unaesthetic situation whereby the

	HP 3000	<b>OA06/13</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

last line of a paragraph becomes the first line of the next column or page. Such an individual line is known in the trade as a "widow-line" - in the case of hot metal type composition it was the practice to reset entire paragraphs, and to insert or delete words in order to avoid widows! Analogous to this, an equally unaesthetic situation can occur in that the first line (or title) of a paragraph stands alone at the end of a column or page. It was with the object of avoiding all such cases that widow-free page make up was developed.

#### **The algorithm of the widow-free makeup**

The widow-free page makeup described here is not intended to replace the block page makeup method already described but instead targets only on that blocks which does not fit into the column as a whole. For the purpose of widow-free page makeup it is assumed that the blocks which are to be made up in accordance with the widow-free page make up algorithm will contain only straight texts i.e. that within this block the line spacing, for example, cannot be changed, and that no text commands may be used which would cause a positive or negative vertical movement. Nevertheless, these limitations make it possible - with the help of a simple algorithm - to make up blocks using the widow-free page making where the length of these blocks may be greater - even several times so - than the defined column length.


With the present integrated system the block-modus-end command is given to indicate how many lines must be at the end or beginning of a column. Initially the number of lines and the natural break position is determined. The natural break position is at the end of that line, which - with respect to the minimum wedges - still just fits into the column, and is described by the number of lines before the column end.

If the number of lines before and after the natural break position is not smaller than required, the split is done at the natural break position. If the number of lines prior to the natural break position is smaller than the required minimum, the block beginning will be transferred to the next column. Should the number of lines after the natural break position be smaller than permitted, an attempt is made to relocate the break position so that lines can be taken over into the next column from the block beginning (i.e. from the lines prior to the natural break position). Should this not be possible without going below the requested minima, then the block beginning is carried over to the next column. Should there be one or more columns between the beginning and the end of the block, the same would apply for the lines up to the first natural break position and from the last break position onwards. The middle part - i.e. that part between the first and the last natural break position need not be considered - "it keeps in step".

The widow-free page makeup algorithm has already been implemented. The development of a more general version - i.e. having less limitations than mentioned above - is featured in the long-range planning.

#### **Conflict situations in widow-free page makeup**

If one or more columns figure between the beginning and end of a block, the intermediate columns will not contain any can-split position at which the column could be whited out. The defined column length can then only be achieved by increasing the line spacing accordingly.

	HP 3000	OA06/14
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

In multi-column directories there is always the danger that adjacent columns may have different line spacing. Since this would give rise to an incongruous impression, in the present system no attempt has been made to provide for the modification of line spacing. Thus it is possible that there will be some free space left at the foot of a column but this will always be less than the spacing between lines.

### **The incorporation of image and graphic in the makeup**

At the moment in the model under consideration images and graphics are incorporated in the data to be made up only by means of an identification key - in the form of a text command. When creating the blocks of the endless column by means of the formatter, the graphics and images are incorporated into the appropriate text block whereby the size of the block - i.e. the size of the graphic or image - is read from the graphic master entry. Accordingly, the size of the encompassing text block may well be identical to that of the graphic or image block. The position of the graphic or the image within a page will be determined by the position of the corresponding text command integrating the graphic or the image.

This means: once the formatter recognizes a "graphic command" during syntax analysis, it will directly access - with the graphic identification key contained in the argument of the graphic command - the graphic master entry in the graphic database and in doing so retrieve the information necessary for the makeup. This assures the actuality of the graphic data.

Pictures and graphics are subsequently interpreted by the formatter as blocks or block parts and are processed accordingly. The data transmitted by the formatter contain - apart from the image or graphic identification key - also informations regarding the absolute (horizontal) position within the page. The proper data of the image or graphic - namely the run-lengths and tone values - do not need to be prepared and incorporated until the transfer (respectively the code conversion) of the made up text to the output device.

At this point I want to remind you again of the enormous storage requirement imposed by real object graphics. The adjacent graphic comprises approximately 86.000 pixels - which in the case of a byte/pixel coding means a storage requirement of approx. 10.700 byte. In the run-length-coded form this same graphic requires only approx. 2.500 byte - or in other words, 76 % less storage space.



### **The problem of the galley-proofs**

Now that the graphics have been successfully digitized, stored, integrated in the makeup, and finally typeset (in theory and test) the requirements become more stringent: with the object of economizing on material and personnel costs, the set ad - including the featured graphics and proper correspondence text (address, order number, customer number, etc.) should be output on normal paper instead of on film or RC paper. This brings us to the complete integration of the commercial (administrative) and technical (production of the composition) applications. For this purpose the laser printer is an obvious choice.

The problem here is that on one hand the types should be identical to those on the typesetter but on the other the laser printer (or electronic printer) resolution is in the region of 300 dots/inch whilst that of the typesetter is in the region of 1000 - 2500 dots/inch. Fortunately there is a solution to this problem. Hitherto it was so that every typesetter or printer incorporated a module which generated the corresponding image-dot matrix from the input type commands and characters - in other words it carried out a scan conversion. With the more modern typesetters (since the laser typesetters became available) this module has been separated from the typesetter. As a discrete unit it is known as the "Raster Image Processor" (RIP).

Some manufacturers have designed their RIPs so that it is possible to control a typesetter or a laser printer as desired whereby the RIP can output the typesetter fonts - loaded into it - in original size on the laser printer. Once it becomes possible to make the RIPs capable of processing graphics and they are fitted with the same scale function for graphics as for types, my galley proof problem will be solved. I can well imagine that it will become possible to use a RIP even for a number of printers of different resolution or that it will become possible to control a high-resolution terminal by means of a RIP.

### **Epilogue**

The system described in this present work was implemented by me as a development version in October 1985 and serves presently almost exclusively for the testing of different hardware items (scanners, printers, typesetter, etc.). On the basis of experience gained so far it is possible to formulate specific requirements expected of a future system whereby only some of the desired hardware components are currently available in the required quality or in an economic quality/price relationship. I trust that these components will be available within the near future and that they will provide a new "key to information".

### **Postscript**

During the preparation of this paper I was informed about the new Hewlett-Packard Desktop Scanner HP 9190A ScanJet. Unfortunately I had not yet the possibility to test the ScanJet, but the technical data as well as the price seem to make it an interesting piece of hardware.



## **Bibliography**

- ABEL86:** Abelow, Gerald R: "Digitized Logos" in: *Interact* Vol. 6, Issue 12, INTEREX, Sunnyvale 1986
- APPE85:** Appelt, Wolfgang: "Konzeption eines Arbeitsplatzes zur Erstellung wissenschaftlicher Texte in hoher typographischer Qualität" in: *Informatik Fachberichte* Bd. 108, Springer-Verlag, Wien 1985
- FOLE84:** Foley, J.D. / Van Dam, A.: *Fundamentals of Interactive Computer Graphics*, Addison-Wesley Publishing Company, Reading - Massachusetts 1984
- GARM86:** Garms, Hans-Joachim: "Reproduktion von Schriften in Computern" in: *Informatik Fachberichte* Bd. 119, Springer-Verlag, Bremen 1986
- KNUT79:** Knuth, Donald E.: *TEX and Metafont - New Directions in Typesetting*, Digital press (USA) 1979
- KNUT84:** Knuth, Donald E.: *The TeXbook*, "American Mathematical Society" und "Addison-Wesley Publishing Company", Reading - Massachusetts 1984
- PURG85:** Purgathofer, Werner: *Graphische Datenverarbeitung*, Springer-Verlag, Wien 1985
- SPRI82:** Springstein, K.-A.: *Die elektronische Bildverarbeitung von A - Z*, Verlag Beruf und Schule, Itzehoe 1982

# Desk Top Publishing

## - our first three years

by Tim Cullis  
HP Computer Users Association

*Desk Top Publishing - The creation, by electronic means, on a desktop computer, of printed matter for publication and for sale.*

### Abstract

In the last nine months, the computer press has been close to hysteria reporting developments in the field of Desk Top Publishing - one wonders if the fascination is due to the subject being so close to the hearts of journalists.

Briefly - DTP can be defined as the ability to produce in-house artwork using desk top computers, normally in conjunction with Laser Printers and probably in conjunction with sophisticated software. According to internal H-P projections, the industry-wide wide market for desktop publishing applications is expected to reach four billion dollars of PC equipment by 1990. New products are being announced by manufacturers every week. So many of these are being marketed before being ready for release (or even half-way through the development) that a new term "vapourware" has been coined.

You may not, however, need the facilities of some of the more esoteric software and hardware packages which are being developed. In the UK Users Group we have been using DTP techniques for the last three years using alternative, less expensive techniques. This paper shows how we have used these methods and covers:

- *A case history of some of the techniques which we have been using in-house.*
- *An introduction to the subject of printing, typesetting and lasersetting with a definition of the terminology you will meet.*
- *An overview of some of the newer software packages and DTP hardware which is becoming available.*
- *Conclusions as to the combination of products which are likely to be cost-effective for differing requirements.*



### Our first experience of publishing

When I started working full time for the HP Computer Users Association (UK Users Group) in January 1984, one of the priorities was to produce a regular news sheet for members.

The first attempt was two pages long and was produced on our newly acquired HP150 using MemoMaker and an extremely slow HP2602A (a 20cps daisy wheel printer). The finished product was photocopied by the owner of the local Indian grocery shop, with whom I had negotiated bulk copying rates. The total production cycle was probably about 12 hours - which has never been bettered since!

### The first real issue

By the time the first issue had to be prepared, I had met a company exhibiting at a trade show who were advertising that they could take floppy discs off any micro-computer and transform the text via a Linotron 202 digital phototypesetter into beautifully set output which could then be pasted up and printed.

I decided to give it a try and the input was again done on the HP150, but then we hit a snag - any micro-computer meant any reasonably standard micro-computer, and our equipment used 3½" discs and theirs used 5¼".

Not to worry - I found out that if we connected the serial port of the HP150 to the serial port of their ACT Sirius micro, we could transfer the data using a communications program. The only drawback was that I had to physically carry the HP150 to their printing works, which were located on the fourth floor of a building without a lift.

The production of the newsletter was not straight forward. Typesetting commands such as

*{m42 f84 h10 111}*

had to be inserted in the text to change the typeface or the weight or size of the character **and if you made a mistake and forgot to change back to body copy definitions after printing a heading, the remainder of the article would be in bold typeface (as with this paragraph).**

Nevertheless I found my first real experience of printing and page layout very exciting and an example of the issue is shown in *figure 1*.

For comparison purposes, *figure 2* shows some of the original text, together with the embedded typesetting commands, which was used in the production.

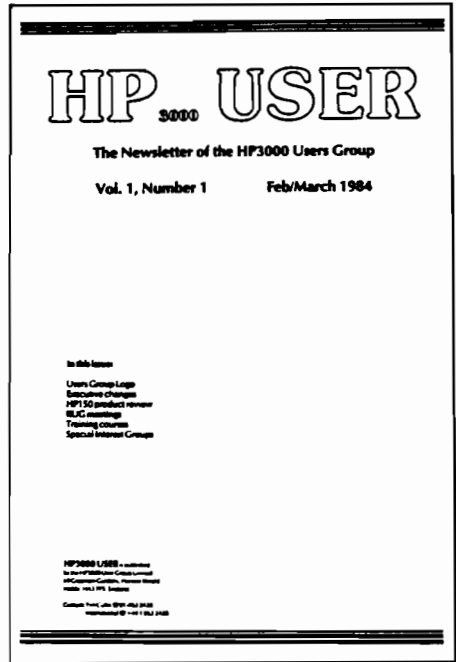


fig 1: HP USER issue #1

```
{m45.6f828h72115} ~XH~AM~AR<
HP{h15}3000{h72} USER~
<
{f88h15i15}The Newsletter of the HP3000 Users Group <
<
Vol. 1, Number 1{f111111111111111111111111}Feb/March 1984~
<
{f88h14i14}In this issue: <
{f86h9i10} Users Group Logo <
Executive Changes <
<
{f88h14i14}HP3000 USER {f86h9i10} is published <
by the HP3000 User Group Limited <
<
Contact: Tim Cullis \-.096\01-863\2428 <
International\-.096\+44\1-863\2428
<
{m16f88h14i14}
```

fig 2: text and typesetter commands

**Page layout**

At the stage it is interesting to look at the design of the first typeset issue - especially the page layout. Most phototypesetter machines produce galleys with a maximum width of about 8". In a traditional environment the paste-up person then takes over and lays out the text together with any photographs, diagrams and illustrations using a basic page layout as a guide. Not knowing anything about the process, I wanted to try to lay the first issue out to find what was involved.

The first step is to design the *thumbnail*. This is a sketch of the page showing where normal text columns lie, where page and running headings are to print etc. The example in *figure 3* shows the thumbnail we currently use for our *HP3000 USER* magazine. The main decision you have to make at this stage is the numbers of columns per page.

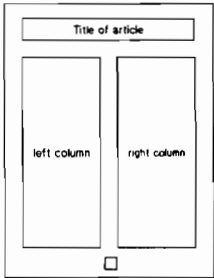


fig 3: two column thumbnail

To a degree the choice depends on the typeface chosen, the point size and the leading, so let's switch subjects to take a look at typefaces.

**Typefaces**

Times New Roman is one of the best known and widely used typestyles. It was designed by Stanley Morison in 1931, initially for the Times of London and was scientifically designed to save space in newspaper printing. Although a compact typeface it is nevertheless very readable.

Interestingly copyright laws provide no protection against the copying of typeface designs, however the typeface's name may be protected by trademark - hence Times Roman is also known as Tempora, English, London, Press Roman and finally, on the LaserJet, as Tms Rmn. Times, Palatino and Souvenir are three

examples of serif fonts. Examples of san serif fonts are Univers and Helvetica. Again, Helvetica is known by other names such as Swiss, and on the LaserJet as Helv.

Within a particular typestyle family are many variations. The thickness or weight of the letter can be varied to produce ultra-light, light, medium, semi-bold, bold etc. The angle of the letters can be altered to produce italics, and the text can be expanded or compressed. Finally, the size of the letters can be varied. The example in *figure 4* is of 17 typefaces in the Univers family, which is a sans serif style.

- Univers Light
- Univers Light Italic*
- Univers Light Condensed
- Univers Light Condensed Italic*
- Univers Medium Expanded
- Univers Medium
- Univers Medium Italic*
- Univers Medium Condensed
- Univers Medium Condensed Italic*
- Univers Bold Expanded
- Univers Bold
- Univers Bold Italic*
- Univers Bold Condensed
- Univers Bold Condensed Italic*
- Univers Extra Bold
- Univers Extra Bold Italic*
- Univers Ultra Bold Exp.

fig 4: Univers typeface family

**Leading**

One of the important printing terms is *leading*. In the days of cold lead typesetting, a leading was a strip of lead inserted between lines of text to give a degree of space. Thus it is normal to talk in terms of 10 on 11 point, which means 10 point type on a 11 point line size (ie 1 point leading). This paragraph is set in 10 point Times on a 10 point line, that is no leading between lines. You will notice that the bottom of the descenders *g*, *y*, and *q* just about touches the top of ascenders *t*, *h*, and *l*.

Most of this paper is set using 10 point on 12 point line size, but for the purpose of comparison, this paragraph is set using 10 on 14. The additional 4 points of leading produces lines which are too far apart for normal text and results in too much white space.



### Basic layout tips

Use a serif font such as Souvenir, Palatin or Times for the body copy. Serif fonts are easier to read and give fullness and ornament. It is interesting to note that the American Interact magazine uses a san serif font. Use a type size of between 9 point and 12 point for the main copy.

Justify the text with a smooth right margin - do not run ragged. Use plenty of visual rests such as the crossheads in this article. The crossheads are normally the same size (or very slightly larger) but in a medium weight. In this paper the crossheads are Times Medium 12 point, but it would be acceptable to use a san serif font such as Helvetica for this.

As a matter of principle, try to use only one serif font and one san serif font per document. This isn't difficult if you are using the LaserJet as there isn't a great variety to choose from! Provide variation by differing sizes and weights.

Pullquotes are also useful for providing a visual rest. Pullquotes are also known as teasers or decks and are provocative phrases or sentences pulled from the body copy and reproduced in a larger typesize.

The column width used in this paper provides space for around 45-55 characters per line, varying according to the mixture of lower and upper case letters. Depending on type size, it is normal to put between 40 and 60 characters per line. Any more than that and the eye has difficulty tracing its way back to the start of the next line. Leading can help on wide columns by introducing some white space to effectively guide the eye back to the left margin.

Having columns which are too narrow is just as problematic. If the text is being run-in as justified (ie smooth right margin), unsightly rivers of white can appear in the text.

This is particularly apparent where exceptionally drawn-out vernacular is used - as in this column. This can be overcome to a degree by using hyphenation but beware of excessive use of this. For example, *major* should not become *ma-jor*.

Some packages have automatic leading - if you choose 8.5 point text it will lay it in a 9 point leading, 10

point text in 11 point leading etc. Switch off auto leading to ensure text blocks align - otherwise you will get uneven lines across columns. You can see this in the triple column example above where the left hand column is on a different leading.

Another printing term you may come across is *widows* and *orphans*. A widow refers to the case where the first line of a paragraph is one one page, the remainder on the next. An orphan is where all the paragraph except the last line fits on the page and the last line has to flow over. Widows and orphans should be removed by judicious editing, forcing page breaks early (widows) or by using hyphenation to place text closer (orphans).

The print industry is a closed shop and it is difficult for an outsider to discover techniques. I wondered for ages how magazines always managed to end their articles at exactly the bottom of the last column of the page. At the beginning of 1985 I took the opportunity to visit the Intertext publications group to pick their brains. I found out however that at the time, although they had nine staff in the group, the entire job of typesetting, graphics design, paste-up and production was subcontracted to an outside agency.

Eventually I found the answer to my question about ending articles at the bottom of the page seemed to be via trial and error.

### Our next printer

Getting rather tired of carrying the HP150 up and down four flights of stairs, I was pleased when I met another printing company, Parchments of Oxford.

As well as being a great bunch of people and eager to initiate an outsider in the intricacies of printing, Parchments owned a *Magic Machine* which could read 8", 5¼" and 3½" discs with about 400 common disc formats, so it was then possible to send the text files on disc rather than lugging the complete HP150 round to the printers. We have used Parchments even since for printing and typesetting.

### Phototypesetters

A phototypesetting machine works by reading text off a floppy disc, looking up the font characteristics which are stored on a hard disc, and using fibre optics, LED or laser technology to paint a beam onto photographic paper. A typical machine operates at 2 million characters per hour.

In addition to the standard alpha-numeric fonts, special characters are normally available. For example a right facing pair of scissors together with a row of dots and a left facing pair of scissors could be used to denote a snip off form. Other special characters may include a telephone symbol, copyright or trade-mark symbols.



### Providing typeset input

Parchment's typesetting machine was a different make to the one we had used before and operated with a different command language. Rather than learn a new set of codes we decided to let Parchments insert all the typesetting commands and we would prepare the magazine input using MemoMaker. Once the input was ready, a disc would be sent to Parchments together with a hardcopy. The hardcopy would be marked up using different colour highlighter pens to denote differing typefaces, font sizes etc.

A variation on this was to use MemoMaker bold and underline characters imbedded in the text to denote bold and italic. Parchments could quite easily do a global search and replace with proper typesetting commands.

We installed a facsimile machine in the office to improve the turnaround time and once the galleys had been produced, a copy would be sent to us over the telephone lines. We would then take the various advertisements and the galleys and do a paste-up of the complete magazine. The paste-up would then be returned to Parchments together with the advertisements and they would proceed with the production side.

### Other printing terminology

A term you may come across, especially when a salesman is lying to you, is *half-tone*. A photograph in a magazine or newspaper is a half tone and if you look carefully you will observe the differing shades of grey are obtained by means of varying the size of the dots in being printed (ie keeping the spacing the same).

Despite what salesmen say, laser printer and scanners are not able to produce half-tones. What they do to try to emulate a half-tone is to keep the size of the dots the same (no choice) but to print them at varying intervals.

You will probably come across the term  *Kerning*. This refers to the need for some combinations of letters to be brought together so that one overhangs the other. A common example would be the letters "To". Many DTP packages provide automatic kerning pairs, some provide additional manual kerning but this is largely bells and whistles

If you come across the terms *EN space* and *EM space*, these are standard width measurements which originally were based on the width of Ns and Ms. Still on measurements, an inch is equivalent to 6 *picas* or 72 *points*. Finally, a *rule* is the printing term for a horizontal or vertical line, which can be of varying width.

### The move to laser setting

We bought our first HP LaserJet in autumn 1984. This was initially used with MemoMaker for simple letter writing and invoices. In January 1985 I obtained a

review copy from HP Santa Clara of MicroSoft Word on the HP150. MS-Word so perfectly matched the capabilities of the laser we decided to purchase a copy.

One of the early laser-set jobs attempted with MS-Word was our *Warwick '85 Conference Programme*. One of the problems with scheduling conferences is last minute speaker changes, and we were able to hold open the programme until ten days before the conference.

#### CONFERENCE WELCOME

Welcome to Warwick'85. This is the second combined conference of the Hewlett-Packard user groups in the UK and the conference Centre of "Computer Systems for Managers and Professionals" a one which is very much in tune with the times, for users of all ranges of HP machines.

All the time slots, except for plenaries, will offer at least one session relevant to "1000" users and at least two sessions relevant to "3000" users. Papers on the use of PCs will be on Tuesday and Wednesday; the Management Stream papers will be Monday and Tuesday.

Of course, the conference is not only about formal meetings. Much of the benefit comes from informal discussions and the lounge and bar facilities at Warwick University should facilitate this. Should any group need a room for a more organised ad hoc meeting, please contact the administration desk.

This conference could not have taken place without the hard work of the volunteer committees (Lead abstractors) and of Tim Cottle, whose efforts have gone far beyond the call of duty. I am deeply grateful to all of these people for their unstinting co-operation.

On behalf of the co-operating User Groups, and of the organising committee, I would like to wish you a very worthwhile, and very enjoyable conference.



Bill Lamplin, Conference Chairman

Bill Lamplin

#### TABLE OF CONTENTS

Conference Welcome	5	Conference at a glance	30
Acknowledgements	7	Warwick campus map	31
General Information	9	Sunday plans and abstracts	26-28
HP Management Round Table	10	Monday plans and abstracts	32-41
List of Exhibitors	12-22	Tuesday plans and abstracts	44-52
Exhibition layout	28	Wednesday plans and abstracts	54-58

fig 5: Warwick '85 Programme

One day, faced with the need at short notice for a one page article for the magazine, I produced it on the LaserJet and then started to wonder just how many people would notice (or care) if we switched over to an in-house production using lasers. The advantage to us was knowing in advance approximately how many pages would be generated by a particular article, and the ability to do changes at short notice.

I wasn't terribly interested in the potential typesetting cost savings as the amounts concerned were not significant and were probably outweighed by the costs of the lasers. We finally decided to switch over completely to laser output towards the end of 1986.



### Using the LaserJet with MS-Word

Microsoft Word is a very powerful Word Processing system with the added advantage of good support for laser printers. The basic principle behind the laser printer support is the use of style sheets which set standard layouts for paragraphs of text.

For example, you may want a basic page layout of two columns separated by .3", with *running-heads* printed 1" from the top alternately left and right justified according to whether the page number is odd or even; running-head text in Times Roman 14 point; section headings in left justified Times 12 point Bold; and the normal text in fully justified 10 point Times Roman with a .2" left indent at the start of the paragraph (*this is the actual layout used in this paper*).

Using MS-Word, the style descriptions are input once together with a two digit style code, and the style code then attached to paragraphs as required. If the central style description is altered, all the associated paragraphs are instantly altered as well.

I am very eager to start using some of the newer DTP packages as MS-Word definitely lacks some facilities. The most immediate is the inability to judge from the screen what the printed page will look like, and the consequential need to do many trial output runs. Nor does it support special characters, shading or tints, or the drawing or boxes and lines (actually there is a way to fool it into printing a horizontal line).

On the other hand it is fine for relatively simple jobs and even some more complex ones. The whole of our Brighton '87 Call For Delegates brochure was produced using the package with some help from Lexisoft's *Spellbinder Desktop Publisher* - to draw the *Conference at a Glance* diagram.

### Using the LaserJet with Spellbinder DTP

Many people will remember using Spellbinder when it was sold by Hewlett-Packard as Word/125. Spellbinder DTP is the same product, with extensions to handle boxes, lines and shading.

It was one of the first DTP programs available on the PC. I saw it at the 1986 SCRUG conference and purchased a copy. We have been using it since, mainly as a single page make-up programme and for this it is quite suitable.

There are some bad points, not least the fact that Spellbinder never was a particularly good word processor. I have an early version of the package and this doesn't support extended characters such as the £, umlauts, accents etc. The biggest drawback to Spellbinder however is the method of defining boxes and lines with .dot commands. This cannot be confused with user friendliness and a sample of the input text is shown in figure 6.

```
.Y 1 240 240 2 0 1 1 21 20 1 35 7
.Box 50 0 725 987 001 0; page outline
.Line 060 032 700 001; horizontal line
.Box 060 084 210 020 1 040; shaded box
.VP 426 + 10; position cursor
Actual text to be printed.....
```

fig 6: Spellbinder input text

For the good points - although Spellbinder doesn't work in WYSIWYG mode, it does have a *Visual* facility which enables you to preview the output on the screen rather than having to print a copy. Also the output quality is very good and figure 7 shows a sample, in this case a job advertisement to be placed in the local newspaper.

**HP COMPUTER USERS  
ASSOCIATION**  
Central Harrow

*High technology organisation moving to light, airy  
offices close to Harrow-on-the Hill station.*

**Non-smoking office**

The HPCUA is an independent professional association representing users of Hewlett-Packard computers. The Association is moving to new offices adjacent to Harrow town centre and has the following vacancies:

**Accountant/Office Services Manager**  
Supervision of computerised bookkeeping activities, preparation of monthly accounts, financial projections, statutory matters. Also responsible for office establishment services.

**Executive Secretary**  
Interested and varied project work as well as normal secretarial duties. Shorthand not necessary.

**Technical Editor**  
To join a team producing two monthly magazines. Creative writing ability is vital, as is a broad understanding of computer technology/trends.

**Seminar/Meetings Co-ordinator**  
Organising all aspects of National Meetings and Training Courses. Also responsible for membership promotion.

*Benefits include 21 days per annum holiday, non-contributory pension scheme and medical insurance.*

Please send CVs to: T.R. Cullis, Executive Director, HPCUA, Trafalgar House, Grenville Place, Mill Hill, NW7 3SA

fig 7: Spellbinder output



### Using a laser for other tasks

We now have three laser printers in the HPCUA offices, a standard LaserJet, a LaserJet Plus and a LaserJet 500 Plus. A combination of *smart switches* and *protocol converters* are employed in order that the lasers can be shared. In the example in *figure 8*, Vectras are connected to a parallel smart switch and thence to a LaserJet 500 Plus. The HP150's are serially connected to another smart switch, which in turn is connected via a protocol converter to the first smart switch. In this way many people using differing machinery can share the same printer.

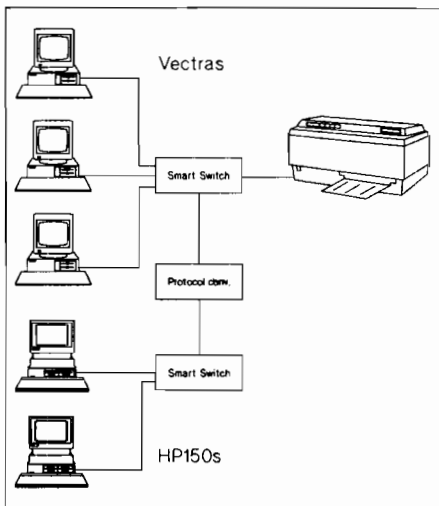


fig 8: Smart Switch Connections

One of the problems sharing a printer is controlling which tray to use, what fonts are in the machine etc. To a degree this can be handled by escape sequences in MemoMaker and MS-Word printer setup strings.

Of course, you can use the printers for many other jobs than DTP and letters. One of our largest print runs was preparing delegate invoices and badges for last year's Brighton '86 conference. Each delegate was set up on our HP3000 based accounting system as a sales ledger account - in this manner we could control receipts and make sure the books were easily audited. A program was written to run down the conference sales ledger, read the variable information and produce two output files, one for invoices and one for badges. The invoice file was a ASCII image of the required invoice

layout, each invoice terminated by a .PA line, which some people will remember is a WordStar end page marker. The file was transferred from the HP3000 to an HP150 using Reflections software, read into MemoMaker and printed as invoices on the LaserJet on standard headed paper.

The badges were produced by downloading the file to the HP150, reading it into MS-Word, globally altering all the text to be 14 point Helvetica and then printing on specially prepared perforated ticket/badge sheets which were hand fed into the manual feed of the laser. Our badge preparation run in total took around two hours for 600 pre-registered attendees.

### How does a Laser printer work?

At this stage we should take a look at how laser printers work. A laser printer consist of two parts: the print engine and the controller. Cheaper laser printers such as the HP LaserJet series use a simple controller, lasers such as the Applewriter use a more powerful form of controller known as a Rasterising Image Processor (RIP). The print engine is borrowed from the world of photocopying and takes care of the paper handling, toner application and fusing; the RIP does the intelligent work in terms of communicating with the host micro and controlling what is printed, and where.

The HP LaserJet is built to HP specifications by Canon of Japan and uses a print engine with a positive image engine. It is the print engine that determines the maximum print speed (pages per minute) and the print quality (dots per inch).

The laser scanners the paper and causes an electrostatic charge where toner is to be deposited. Some other manufacturers use a negative image charge where the whole page is charged and the laser then effectively neutralises the white area. The second method (negative image) produces more consistent and darker print with less tendency to graininess.

The RIP consists of the software and electronics that converts the data sent by the micro-computer into a stream of instructions to control the laser beam. As the printing process is handled by the print engine, the RIP has to buffer the information coming down the line, work out the format and convert it into dots before sending the entire page to the print engine. The print engine may be capable of, say, 8 pages per minute but with complex tasks it may take the RIP several minutes to generate a single page of output.

The HP LaserJet doesn't have a full RIP and the method used to convert the characters into dot patterns is to look up the letter in a font table and pick up the *bit pattern* - the actual pattern of dots which represent the individual letter. The font table can be stored in the ROM of the printer as an in-built font, in a ROM plug-





in cartridge, or in printer RAM as a downloaded font. Each size in a particular font has to be held as a different bit-map. For example, on the TmsRmn/Helv cartridge, Times 8 point, Times 10 point, 10 point italic and 10 point bold are all stored as separate bit-maps. The advantage of this is the fast speed of access, the disadvantage is that you quickly run out of space.

To illustrate the point, the figure 9 shows the memory requirements in bytes for downloadable Times Roman fonts (p/n 33412AD).

**TmsRmn downloadable fonts**

Point	Roman	Bold	Italic
6	15240	14920	15240
8	18310	18690	18880
10	24710	24710	25160
12	30720	30600	32640
14	38660	38920	41280
18		56900	
24		91140	
30		134020	

fig 9: font memory sizes

If you tried to download all the Times Roman fonts shown above you would need .65mb memory and the maximum user memory is .39mb. It should be pointed out, however, the table above only shows the portrait-orientated Times Roman base set fonts. A similar amount of space would have to be set aside for landscaped-orientated fonts (increasing the total to 1.3mb).

The base set downloadable fonts also include Helvetica in portrait and landscape styles (total now 2.66mb) and additionally the supplementary set of downloadable fonts fill in the gaps in point sizes, giving a total download of 5.8mb. Not having quite that amount of memory, my typical download is Times 8 point roman, 10 point all (roman + bold + italic), 12 point all, 18 point bold and 24 point bold. This takes .33mb of memory out of the .39mb available.

Assuming you did have the space for a 5.8mb download, all that one has to show it is the ability to print Times Roman and Helvetica in sizes ranging from 6pt to 30pt, either horizontally or vertically - just two fonts out of the thousands available using proper typesetting machines.

Current downloadable fonts are limited to 30 point - although the LaserJet Series II will take up to 48 point. Soft fonts are downloaded to the LaserJet by copying via the serial or parallel printer interface (if possible, use parallel - it's much faster. If you have to use the serial

interface it's worthwhile altering the dip switches to bump up the speed from the factory-set 9600bps to 19200bps).

It should also be pointed out that a 24 point typeface takes 3 times the space as 12 point, and also 3 x the time to download. ASCII fonts take up much less space than Roman-8 as they only consist of 128 characters instead of 256. One possibility which I haven't followed up is modifying the ASCII fonts to switch the £ sign for the # and only downloading the modified ASCII set.

One point I didn't realise until too late is that the more fonts you download, the less room left for graphics. The reason H-P put so little memory in the original LaserJet ranks high as one of the mysteries of the computer world (closely followed by why they still didn't get it right on the LaserJet Plus which really needs three times the memory it has).

The Laser can also print graphics and the normal graphics resolution of an HP LaserJet is 300 dots per inch. Each dot is represented by a bit which is either on or off, so one square inch requires 300 x 300 bits, or 11k of memory. An A4 page of 8" x 11" requires about 1mb of memory. Neither the standard LaserJet nor the LaserJet Plus have enough memory for a full page 300 dpi dump. The LaserJet Plus can just about manage a page at 150 dpi - provided there aren't any downloadable fonts in memory; the standard LaserJet can only manage a page a 75 dpi - which in turn only requires about 64kb memory.

**LaserJet Series II**

This will be introduced at the beginning of March and is a second generation LaserJet. Still based on the 300 dpi Canon engine, it is substantially smaller (8.5" high versus 11.4"), lighter (50lbs v 71) and with a different toner cartridge design which is stated to give blacker blacks and 4000 copies per cartridge (v 3000). It only takes 30 seconds to warmup (compared to 2 minutes) and takes paper weight up to 35lbs.

The standard machine comes with 512kb memory and is LaserJet Plus compatible. Six internal fonts are provided (no great deal - merely portrait and landscape versions of Courier 12pt, Courier 12pt Bold and Line Printer 8.5pt). Maximum character height is increased from the 30 point LaserJet Plus to over 500 point. Two standard LaserJet cartridge slots are provided at the front of the printer and one expansion memory slot is provided which can take either 1mb, 2mb or 4mb memory (giving 1.5, 2.5 or 4.5mb in total). The basic printer will be around £2600 with additional memory priced around £400/mb.

As well as serial and parallel, interfaces are available for ESI ShareSpool, DDL video and Apple Talk. To say the LaserJet II will be successful is an understatement - get your orders in now.

## LaserJet 2000

Another new machine appearing at the beginning of March, the LaserJet 2000 is aimed at networked systems and minicomputers. It prints at 20 ppm, and is compatible with the LaserJet Plus software, formatting and font cartridges. Double sided printing is optional and the maximum resolution is 300 dpi. Supported papers sizes include A4 and A3 and a 1500 sheet correct-order face-down output hopper is standard.

There are 34 resident fonts (*I hope this doesn't mean 34 variations of Courier and Line Printer!*) and the standard memory is 1.5mb expandable using \$750 1mb boards to 5.5mb. Two cartridge slots are at the back of the printer.

Looking suspiciously like a Canon photocopier and standing 4ft high, the LaserJet 2000 comes in three models: 2684A at \$20500 is the base machine with twin 250 sheet input trays, 2684P at \$22000 adds a 2000 sheet input paper deck, 2684D at \$25500 also adds a duplex area underneath.

One facility with an A3 device such as the LaserJet 2000 is to print original pages at A3 size using, say, 14 point text and then have the print shop photographically reduce it to A4 for printing. This technique effectively more than doubles the resolution.

## Apple Laserwriter

Many of the early packages were based on the Apple Macintosh. The Mac has a bit-mapped graphics display which is ideal for the display of mixed text and graphics. Until the advent of Windows and high resolution monitors, the IBM PC and clones such as the Vectra were not suitable for some of the more sophisticated packages.

The Apple LaserWriter printer is actually the most powerful computer built by Apple. Based on the Canon engine, the internal RIP (Raster Image Processor) uses a 12 MHz Motorola 68000 CPU with .5mb ROM and 1.5mb RAM memory. The RIP uses a language called Postscript - *more about this later* - which takes a master outline of a character and scales it up and down to obtain the varying point sizes.

The LaserWriter is often criticised for is slow speed, especially producing the first page. Postscript stores fonts in outline form and before a character can be printed, its outline must be converted into a bit map. The conversion takes about 0.01 seconds for a standard character in ordinary text and substantially longer for large or complex characters.

Postscript saves each character's bit map in a cache to avoid repeating the conversion every time it prints a character. The time consuming conversion process is why the first page or two take longer to print. The author's of Postscript, Adobe Systems, use a document

of 3500 characters in 12-point Times in its speed test, but the test is performed after the characters are cached in the printer's memory.

Although fine in theory, scaleable fonts will not provide the same finish as non-scaled fonts. *Figure 10* shows a 6 point and a 18 point sample of metal typeset output, both scaled to the same size. You can see that the 6 point type has darker stems and bowls, wider letter shapes and looser space between letters. This is in order to improve readability of the smaller font. Scaling all point sizes from one master outline will not provide such features.



fig 10: scaled metal type

## Page Description Languages

We now get onto the subject of Page Description Languages. HP's standard in this area is called PCL and has 4 levels defined, allowing programs which have been written for, say, a level 2 printer to be run on a level 3 printer. The LaserJet sits on one of the top levels, using a superset of escape sequences.

Canon-badged lasers use Diablo escape sequences - which for the purpose of this article can also be classed as a PDL. Adobe's Postscript is used on the Apple LaserWriter and is a very powerful language capable of drawing circles, rotating text in a spiral, shading etc. Postscript is a non-compatible subset developed by ex-Xerox employees from Xerox's Interpress. Other PDLs include C-script which is a clone of Postscript, Chelgraph's ACE, Prescribe which is used on the Kyocera laser, and Xerox Interpress.

## DDL versus Postscript

Hewlett-Packard recently announced they would be introducing a new PDL called DDL on the LaserJet family. This is to be licensed by HP from IMAGEN Corporation. According to HP, DDL is better for longer documents than Postscript and is a second generation product. Adobe System's licencing scheme for Postscript may have had something to do with HP's decision - they insist on 10% of the printer wholesale price of each machine sold.



The DDL system from HP will be largely external to the printer. Based on a 68000 processor with 2mb memory, the interface card fits into an expansion slot on the Vectra and is then connected to the LaserJet by a high speed video interface. The card provides in-built scaleable fonts to match those available in the LaserWriter. Both DDL and PCL standards can exist side by side in the LaserJet. The Video Interface is similar to that on the LP300 laser launched by Cordata at Comdex '85 (1.89 MHz transfer rate).

In the meantime, rumours abound that HP's DDL project has hit delays and won't be available now until mid 1987. In the meantime, one of the options open is to retrofit a Postscript RIP to a standard LaserJet. I am aware of two such systems, but one aspect which isn't published is that you then lose PCL support and all fonts, whether downloadable or cartridge. PS Jet replaces the top cover assembly - and provides 13 typefaces in any point size for \$2995. JLASER PLUS from Tall Tree Systems adds a 600 x 300 dpi resolution.

Another alternative is the LaserMaster TurboCharger. This is an add-in board for a Vectra or IBM-PC compatible which works in conjunction with an HP LaserJet or Canon-based printer to speed up graphics output. It can be used with Ventura Publisher, MicroSoft Word, WordStar 2000, Spellbinder Desktop Publisher etc. The board has 1.5mb memory and a vector to raster converter with a video interface. 60 fonts (variations) are standard. The board costs £1395.

### Other Laser Printers

The Kyocera is a 10 ppm engine, also found wearing Mannsman Tally or Sperry 37 badges. Advertising material suggests it is LJ+ plug-compatible, but the emulation is incomplete - it can't do graphics in landscape and portrait graphics are rather ragged. It has three dynamic fonts - Times, Helv and Gothic, and is driven by Prescribe PCL.

The Dataproducts LZR 2665 is Postscript A3 sized laser capable of 26 ppm output. Uses negative image techniques. Price around £15,000

Norbain Data Systems market what is currently the fastest (?) desktop laser printer. Costing around £26,500 it is based on the Hitachi engine, is capable of 40 ppm and uses two Inmos 32-bit Transputers as controllers.

Agfa P400PS is a 400 dpi Postscript compatible LED page printer capable of 18 pages per minute. Uses the Adobe RIP with a 20mb hard disc to hold downloaded fonts. Up to 6mb RAM to pipeline page processing. Optional 2000 sheet paper bin and 20 bin collator.

### Lower cost typesetting machines

The Linotron 100 is a laser typesetter from Linotype Limited of Wembley (who are actually an

HP3000 user). As well as producing standard galley type output in proof or high resolution mode, it can also produce full page A4 in positive or negative, right or wrong reading. Font sizes from 4 point to 127 point.

The Monotype Blaser is another low cost typesetter with text sizes ranging from 6 to 96 point. Interfaces to the Mac. From Monotype of Redhill.

### Desk Top Publishing Packages

Now we get on to all the vapourware! For a start, let's set the scene about what a DTP package is supposed to do. DTP packages automate the paste-up process. If you have never had the need to do a manual paste-up you are unlikely to require something like PageMaker. They are NOT word processing packages. You can't design a great page layout programme and expect it to be a great word processor as well. The design goals are completely different - word processing should handle very long documents, scroll very fast and include features such as outlining, merging, search & replace. Page layout has to concentrate on production of the printed page. And in the same way that word processing programs do not write, page make-up programs do not design pages for you.

Windows and GEM are the key to DTP flexibility. There are now many different packages, different monitors, different printers, not to mention the variety of page description languages. Windows allows you to custom configure these and also to cut and paste between application packages. The normal method of working which a DTP section might well adopt is for several stations to be used for text input, using something like MS-Word with simple style sheets and variable sized text, and for a central station to be used for the actual task of on-screen paste-up.

Try to match the DTP package and the WP package so that simple textual highlighting can be carried across during the cut-and-paste, rather than having to re-apply the highlighting from within the DTP package. Another consideration may be to choose a package which can output to a phototypesetting machine as well as a laser printer.

The cost of the software packages is not a significant amount of the total expenditure. If you have diverse needs, you may choose to purchase more than one package - but be aware of the training overhead.

### Pagemaker

The leading DTP system on the Apple Macintosh, by Aldus Corporation. Currently being totally rewritten in C and to be implemented simultaneously on the Mac and IBM PC/AT. Earlier version suffered from being too page orientated - not really suitable for a document larger than 8 pages. New version can theoretically



support 128 pages per file, with page sizes from A5 through to A3 and Tabloid.

Pagemaker's early lead on the Mac does not necessarily mean it will be the winner on the IBM PC and compatibles. Has the advantage of being well supported and good manuals. Suitable for small newsletters and short documents. Each page has to be initially copyflowed separately - which is rather annoying, but changes thereafter will automatically ripple through the document. A variety of pre-determined templates, including 21 newsletter designs, are being developed, to be sold as the PageMaker Portfolio Series.

### Xerox Documentor

Purpose built, non-IBM compatible hardware comprising Xerox 6085 workstation plus Xerox 4045 10 ppm laser and Viewpoint software. Good on-screen resolution of 1184 x 925 pixels, however totally spoilt by only supporting text up to 36 point!

Users are advised not to turn off the system as it takes 25 minutes to re-boot. The Xerox 4045 laser uses Interpress PDL (with poor definition fonts) and output can be re-directed to a Compugraphic phototypesetter provided it has an Interpress interface. Prices from £7500 to £14000 depending on size of screen and capacity of hard disc.

### Ventura Publisher

Marketed by Xerox. Runs on IBM-compatible machines under the GEM environment. Allows you to determine where pages and columns start and comes with its own style sheets. Never actually places the text on the page, but merely tags the appropriate files, rather What You See Is What You Will Get. Good for manuals and supports up to 64 chapters, each of around 50 pages.

Requires a lot of disc accesses and would benefit from using extended memory as a RAM disc. Drives Postscript lasers and typesetters, and HP Lasers with Interpress support in the pipeline. Probably the best overall, although Xerox hasn't got the distribution act together to push it as much as it deserves (spending all their time training the sales force to sell the Xerox Documentor). Costs around £795.

### Other DTP packages

**Frontpage:** From Studio Software, sold in UK by Paintpot Computers. Costs around £750, or £1300-£1800 with typesetter interfaces. Drives a variety of Lasers and supposed to have both PCL and DDL drivers.

**LetraPage (MacPublisher on Mac):** Developed by Boston Software on the Mac, the rights to MacPublisher

were taken over by Letraset and the package renamed LetraPage. Due to development problems, the package was dropped in mid-February - an example of vapourware!

**Ready-Set-Go:** The second most popular DTP package on the Mac, Ready-Set-Go was developed on the Mac by Manhattan Graphics and marketed in the UK by Heyden. In mid-February in an amicable agreement, the marketing rights were transferred to Letraset (see above). Similar to Pagemaker in concept - does some things better, some things worse.

**SuperPage II:** BestInfo's SuperPage runs on IBM PCs and outputs to 25 typesetters and laser printers. Can be run standalone or networked via Novell's Netware LAN. Used predominantly in a multi-user high production environment.

**Harvard Professional Publisher:** Again from BestInfo. Has its own front-end - doesn't run under Windows or GEM. Is a compromise between Ventura and Pagemaker. Also quite complicated.

**Wordcraft Elite and Image-master:** Developed by Wordcraft of Colchester, sold also by Canon as part of their Personal Publishing System. Powerful WP system with page makeup bolted on.

**Autopage:** An add-on to AutoCAD, AutoPage allows text pages from Wordstar and AutoCAD drawings to be superimposed on a page. Useful for technical manuals - and there are 50,000 AutoCAD users world-wide.

### High-end systems

There are quite a few high-end systems which shouldn't concern most users. DialText from Talbot Computers based on Macs with Newswrite software. One installation is in Eddy Shah's Messenger Group of newspapers where 98 Macs are set-up in networks of up to 14 machines. Other such systems include ATEX - used for the new *Independent* newspaper, Interleaf, Textet, Omnipage and Xyvision.

### Design aspects

Page composition is as much an art as a task, requiring a new set of skills and a different working vocabulary. While virtually anyone can use a word processing package, designing an attractive page is not necessarily an easy task. Users who do not understand the aesthetic restraints imposed by the rules of typography are easily tempted by the multitude of fonts and typesetting tricks offered by the programs. The result can be a mess - documents that communicate less instead of more.

Consider the difficulties involved in publishing your own documentation. You need someone who can plan a layout, make information accessible, write coherently,

spell, proof read, have design skills, an eye for a diagram, and the patience to put a product together. They must be happy working with leading-edge technology and have more than a passing knowledge of magazine production through old channels if they are to appreciate the new. In addition they need the time to put together company memos, reports and promotional material - and to stand by a laser printer whilst the pages trundle off at somewhat less than high speed.

There are probably only 100,000 to 200,000 people in the whole world who have the ability to produce good looking type. If the DTP market is to be as large as predicted, the only way is either to train everyone to be a typesetter, or to isolate the user from design decisions by using pre-set styles and formats. Some programs attempt to overcome this by setting up strict style-sheets which force the use into a sort of straightjacket. While this attitude helps, more documents will end up looking the same which rather defeats the purpose.

What is really needed, even in a smallish company is dedicated personnel. With such requirements it's probably better to stick to your local typesetter. Sellers of DTP packages argue that buyers will save money by dispensing with expensive outside designers. But graphic design requires aptitude and professional skills which have to be learnt - and the learning process is expensive.

The price for a complete DTP system is about £8000 to £10000 (\$11000 to \$14000) and this is too much to pay for machinery which is only used occasionally. Beware of costings which show the equipment cost being defrayed over five years - we all should know that three years is more sensible, and over three years some of the sums look terrible. Publishing programs that run on existing PCs have not helped because most of the computers need extra hardware such as high-resolution screens or a mouse, and an expensive laser printer.

### Technical Summary

Talking about equipment, what is needed to set-up a DTP centre? Hewlett-Packard's answer is the new Vectra Publisher bundled, no doubt, with a LaserJet II. The Vectra Publisher is a Vectra Series 50 (640kb memory, serial/parallel port, 1.2mb floppy, 20mb hard disc), coupled with a Mouse, PageMaker and MS-Windows software and either an EGA colour monitor or a Hercules-spec monochrome.

I think HP should be more realistic about the type of hardware needed for DTP - anyone buying the HP bundle will quickly run out of steam. For a start, 20mb disc space won't go far with Graphics Gallery raster files taking up 1mb per A4 page! An HP 40mb disc would be more sensible - as well as being a good deal faster. It may be worthwhile investigating larger discs - 80mb+.

To extend speed up disc accessing times, buy lots of extended memory and run it as a RAM disc.

If you value your eyesight, don't buy an EGA monitor (640 x 350 resolution) or the HP monochrome (730 x 348). Instead choose a Wyse 700 (1280 x 800) which is about the same price as HP's EGA and three times the resolution. Other more expensive alternatives are the Moniterm Viking II (1280 x 960) sold by Riva and Sigma Design's Laser View monitor (1664 x 1220). If you like upright screens, look at the Genius Full Page Display System, or the Etap A4 display (720 x 1456) sold by CCA Micro Rentals. Finally, if you've just won the pools look at the Conographic 2800 which has a resolution of 2880 x 1024 pixels. An optional ConoVision 2800 RIP can drive the HP LaserJet at 600 x 300 dpi resolution, with a format time of 8 seconds per page! Finally, on the laser side, by all means buy the LaserJet II, but make sure you add at least 2mb and preferably 4mb of extra memory (don't forget - there is only one slot so if 2mb isn't enough you can't just add another).

### Do you need a scanner?

Another announcement at the beginning of March is the HP ScanJet. This is a 300 dpi scanner with a price that should wipe the other systems off the market.

But do you really need a scanner? Why take a perfectly good diagram or photograph into a scanner and then reproduce it at only 300 dpi. Bearing in mind the problems of half tones, the best way to paste in photographs and diagrams is manually before the page is filmed for printing. Do not do something just because it is possible. The only viable uses I can see for a scanner is when the document being prepared is going to be printed as originals on the laser, when the scanned picture is to be transmitted as part of an E-Mail document, or possibly, in a high volume shop, for placement purposes only.

### Summary

Once some of the mist clears from all the vapourware, we will probably see some very good DTP packages. For someone in the print trade it is a very exciting time, and a race between laser phototypesetter technology getting cheaper on the one hand, and laser printer technology getting higher quality on the other. The main problem on laser printers at the moment is the extremely poor choice of fonts available.

Research and development of 300 dpi technology is finished, most people are looking towards 480 dpi (new AppleWriter), and beyond to 800 dpi+. Colour lasers are on the horizon using the four primary colours - yellow, cyan, magenta and black. The main problem is one of registration (getting printed dots from each



engine to line up). QMS are actively developing a four colour laser with a product announcement expected soon.

Look to see new industries springing up to support DTP - high street laser copier centres, designer layout packages, font families etc.

### A brief history of printing

The necessary components of printing (the art of paper-making, the invention of ink made from lamp-black, and crude wooden and stone seals) came together in China around the fifth century AD. It took another thousand years for paper-making and relief printing to reach Europe via Morocco and the Arab world.

Nine hundred years later, in the middle of the fifteenth century, Gutenberg invented the letter-press using moveable type. The typewriter was developed just 114 years ago in 1873, followed three years later by the idea of casting type one line at a time (Linotype), followed shortly after by Monotype - where each character is cast separately for higher quality books etc.

The next really major development as far as our interest in DTP is concerned was Xerox's invention of photocopying in 1948, followed shortly after by phototypesetters. These used character masters in the form of photographic strips in conjunction with a flash lamp and some sort of lens. The final invention was the desktop laser printer which was produced by Canon's marrying of copier technology with a digital laser setter.

*Tim Cullis*

### Suggested Reading List

Desktop Publisher (monthly newsletter)  
£125 for twelve issues  
Published by The Desktop Publishing Company

Desktop Publishing Today  
£96 for twelve issues  
Published by Compass Press

The Complete Guide to Pasteup (2nd edition)  
£12.50 by Walter B. Graham  
Published by Popular Communications

Graphic Ideas Notebook  
£13.50 by Jan V. White  
Published by Popular Communications

Editing for Print  
by McDonald  
Published by Porto Publishing ISBN 0355 10787 6

Postscript Language Reference Manual  
£21.95 by Adobe Systems  
Published by Addison Wesley ISBN 0201 10174 2

Postscript Tutorial and Cookbook  
£16.30 by Adobe Systems  
Published by Addison Wesley ISBN 0201 10179 3

The Alternative Printing Handbook  
£4.95 by Chris Treweek and Jonathan Zeitlyn  
Published by Penguin Books ISBN 0 14 046 509 X

The Print Book  
£9.95 by Information Transfer  
Published by National Extension College ISBN 0 86082 666 X

The Complete Guide to Illustration and Design  
Terence Dalley  
Published by Phaidon Press ISBN 0 7148 2347 3

The Illustrated Handbook of Desktop Publishing and Typesetting  
£23.60 by Micheal Kleper  
Published by Tab Books (available April 1987)

Desktop Publishing using PageMaker on the Macintosh  
£14.50 by Andrew Lucas  
Published by Ellis Horwood

The Desktop Publishing Companion  
£14.95 by Graham Jones  
Published by Sigma Press (available April 1987)

Personal Publishing with the Macintosh  
\$24.95 by Terry Ulick  
Published by Hayden

Desktop Publishing From A to Z  
£17.95 by Grout, Athanasopoulos and Kutlin  
Published by McGraw Hill

### Biography

Originally an Accountant before becoming a convert to the high salaries of computing, Tim Cullis' first experience of HP3000 equipment was in 1978 when, as Technical Director of Air Call Computer Systems (a UK software house) he initiated an OEM contract with Hewlett-Packard.

A long time supporter of user groups, Tim was the Programme Chairman for the Edinburgh '83 HPIUG International Conference. Since January 1984 he has worked full time for the HPCUA (UK HP user group) as its Executive Director, starting with a staff of one - himself - working from his study, currently with a staff of eight people, with offices in North West London.

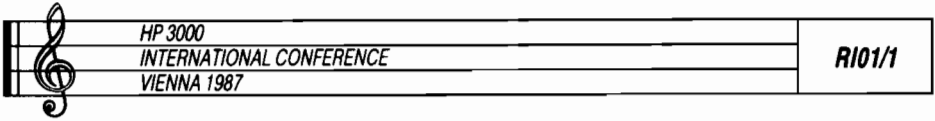
The HPCUA publish two HP USER magazines which are widely circulated in the UK and elsewhere in Europe. Extensive use has been made of desktop publishing techniques since March 1984.



*HP 3000*

*INTERNATIONAL CONFERENCE*

*VIENNA 1987*




## **A Spectrum strategy for the current HP3000 user**

**Nick M. Demos**

**Performance Software Group  
12 Hillview Drive  
Baltimore, MD 21228  
301-242-6777**



	HP 3000	<b>RI01/2</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## A SPECTRUM STRATEGY FOR THE CURRENT HP 3000 USER

The writer was a Systems Engineer with IBM during the introduction of the S/360. The S/360 was a single architecture, implemented in different CPU's, that replaced the existing widely different models of IBM computers. While it established a new standard for all IBM computers, it was radically different in architecture (instruction set) than any of the existing IBM computers (1400 series, 1620, 7040, 7070, 7090, etc.). The parallel to HP's Spectrum project are striking.

The Wall Street Journal labelled Spectrum "the most adventurous gamble the company has ever undertaken." We have all been hearing about Spectrum for a long time. Remember when it replaced "Vision" as the HP 32 bit entrant?

Now that the first of the Spectrum series are about to be delivered, we can take a look at what the initial results of this five year development have been. The first of the 900 series (which is what HP is calling the Spectrum offerings in the commercial marketplace) is the 930. The Series 930 uses Schottky TTL logic to deliver CPU performance of 4.5 million instructions per seconds. It will support 24 megabytes of main memory and has dual I/O busses as a standard feature. It will be the first machine of this series to reach the customer. First shipments of the 930 are expected during the fourth quarter of 1986.

The 930 is a stop gap machine. I would almost say that it is a pilot system, which will soon be replaced as HP perfects its

NMOSIII VLSI chips and other more advanced circuitery. It will incorporate the logical architecture of Spectrum. Its main benefit will be to those customers who have saturated their current model 68's or 70's and cannot divide the application conveniently between two or more CPU's. In order to take advantage of the improved performance, they will have to recompile into native mode and convert to Turbo-Image if they have not already done so. This means that their source programs must be either in COBOLII, Pascal or Fortran 77. Those are the only native mode compilers that are available at first release of the 930. 930 system prices start at \$225,000.

The series 950 is the first of the Spectrum series to take advantage of HP's NMOSIII VLSI processor. It will execute 6.7 million instructions per second. It will have up to 64 magabytes of main memory. It is the first HP product to marry the new architecture with new chip technology. However, it will not be on the price list until later this year. Its' price is expected to be in the \$300,000 to \$350,000 range. This will offer a much better price performance package up grade for current 70 model users.


HP has taken the concept of Reduced Instruction Set Computing, adopted it and added other new developments in hardware and software design. They call the result HP Precision Architecture. The basic features of Precision Architecture are:

1. Reduced Instruction Set
2. Fixed-length and fixed format instructions

3. Load/Store design
4. Hardwired instructions
5. Single Cycle operation
6. Optimizing Compilers.

Reduced Instruction Set Computers (RISC) are the current darlings of the computer industry. In a nutshell, this engineering philosophy states that a well chosen simple set of instructions that can be hard wired is better than a much richer more complex set, usually implemented in microcode to keep the cost reasonable. First of all the CPU is much easier to engineer and therefore can be designed economically to run at much higher speeds. Secondly, well designed "optimizing" compilers can be more effectively implemented for a RISC machine than the previous CISC (Complex Instructions Set Computers) because the compiler writers have fewer instructions from which to choose. The rationale for RISC is that changes in circuit technology, the speed and cost of accessing memory and better high level language compilers have made the CISC machine technologically obsolete. Memory access speeds have increased faster than CPU speeds. RISC, with fixed length, fixed format instructions and single cycle operation is inherently very fast for most operations that have to be performed on today's computers. However, what is "Reduced" is not defined. There are about 127 instructions in the 900. I can remember computers that had only 1/5 that many instructions. The HP implementation of RISC is not that "reduced."

Other elements of the 900 Series are also a radical departure from current 3000's. A virtual memory addressing scheme is used

	HP 3000	R101/5
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

whereby disc memory is regarded as an extension of main memory and is "mapped" into main memory as required. This scheme fits very well into the caching scheme used by the 900 Series and is a very effective method to access both data and instructions. However, it is a radical departure from the current methods, although it eliminates the dual buffering inherent in a cached 3000 system today. Obviously, the entire I/O system must be re-written to support this I/O method.

The software is the critical element in the 900 Series. Not only is a huge amount of software required for the operating system, which goes under the name MPE/XL, but the new optimizing compilers require an even higher level of technological competence than previous compilers. At the same time HP is attempting to integrate their IMAGE database system with their new SQL relational data base. On top of that they are committed to run almost all existing code in compatibility mode. All this is a huge undertaking. Just doing the new XL version of MPE is a larger undertaking than HP has accomplished recently. It is the software area that caused many people to have doubts about HP's ability to deliver the 900 Series on schedule, with reliable software that gives the promised performance. There is no question that HP will be able at some point to deliver a quality 900 Series product with a reliable, complete set of program products, operating at the promised performance levels. However, today they are a long way from this goal. Getting there is no easy matter.

As an interim step to gaining improved throughput on the 900 Series without having to recompile, HP has something called the Object Code Translator. This will gain an estimated 15% speed improvement, and is meant to assist users in cases where they cannot recompile. If it is reliable, it will be of use to users with SPL code.

What does the market think of this radical new product line from HP? There have been several comments that indicate that while the 900 Series is impressive, it has been a long time coming. "HP is currently comparing Spectrum to (DEC's) 8600. By the time the systems come out, they will be competing with an entirely new machine," said Michael Murphy, co-editor of the "California Technology Stock Letter." In many comparisons of price that HP is using to show the advantage of the 930 and 950 over other vendors offerings, HP is including databases as part of the comparison. This makes HP look more favorable because the HP price includes HPIMAGE, while on the other machines the data base is a separate, relatively high priced product. While one might argue that today a data base system is a necessary part of any viable computer system, including it in the comparison can be misleading, because the users of the competing equipment may have less expensive data bases available to them.

HP's success with this new series seems to depend on two factors.

1. How long will it take HP to stabilize the new product at a high level of performance and with a complete software offering? If this can be done relatively quickly and as promised, then HP has laid the groundwork for an architecture

that will benefit them for years to come. If they are slow to implement their excellent strategy, their credibility will be damaged.

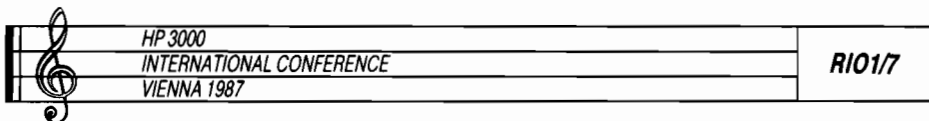
2. What will the competition do in the next year or two to counter and perhaps improve on these new offerings? HP has no lock on any of this new technology. It is known that every other major vendor is looking seriously at RISC and other new technologies to improve price performance.

Where does this leave the HP 3000 user who is looking at Spectrum as an upgrade for his system? The series 70 will be a better choice for many at least in the short term. There are two classes of 3000 users who have an immediate interest in the 900 series.

1. Those users who need CPU power beyond that available on the model 70 and require that it be on one computer (because of the size of a database, perhaps).
2. Those in large corporation with many 3000's who can afford to bring a 900 series in house as prototype for future systems.

Those who are not in one or both of the above categories should probably upgrade to or acquire additional model 70's.


Having made the decision to go to a 930, the user must develop a detailed migration strategy. He must balance the conversion effort of going to native mode with the benefits. Only native mode processing will give him cost/performance benefits that



justify the acquisition of a 930. Even programs to be run in compatibility mode need to be tested at least until there is some confidence that compatibility mode is bug free.

By adopting a radically new architecture, HP has taken a giant step toward rationalizing their computer line for future growth. It is an excellent, bold, decisive strategy on HP's part to increase their market share in the long run. The immediate problem for the current HP user who needs more processing power is how to get from here to there. It will be a while before the 900 series stabilizes into the kind of super reliable system we have come to expect from HP.

The next few months will be very interesting for all of us involved with HP.

	HP 3000	RI02/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## MPE XL Contributions to HP3000 System Availability

Dave Trout  
Hewlett-Packard Company  
2 Choke Cherry Road  
Rockville, MD 20850 USA

### Introduction

This paper will examine new features and enhancements in MPE XL which contribute to increased reliability and availability of the HP3000 900 Series systems. Particular emphasis will be placed on new file system features, the system directory design, Volume Management, and MPE XL Backup/Recovery.

The ability to withstand soft disc failures in MPE XL is a major improvement over earlier versions of MPE and is a result of the enhanced file system and distributed system directory. The new directory design will also allow more I/O concurrency and therefore better performance. Directory expansion is now simple and automatic; a start from scratch (RELOAD) is no longer necessary. Better file space utilization will result from the improved extent allocation in the file system. These are just a few examples of improvements in MPE XL which contribute to a more reliable and available HP3000 system.

Each enhancement will be presented by examining the implementation used and the resulting benefits. Although some details of file label and directory data structures will be given, this will not be a discourse on MPE XL internals. Some features discussed here will not be available in the first release of MPE XL, but are planned for the product. This will be noted where appropriate.

### Background

HP's commercial customer base has increased in diversity over the years. With the introduction of the Series 37 and now the MICRO/3000, we have expanded the number of low-end customers substantially. At the same time, our high-end customers (many of whom are now running large Series 70 configurations) have encountered continuing growth in applications and the resulting need for more computing power. In addition, HP is finding new customers whose computing workload will require more capacity and performance than we have traditionally made available in the past.

In the beginning stages of development on MPE XL, it was very obvious that HP must address these growth trends on both ends. Three very important objectives for MPE XL were defined: 1) greater performance and capacity, 2) enhanced ease of use and functionality, and 3) higher availability and reliability. To our high-end customers especially, the greater performance and capacity is an obvious requirement. But equally important is the higher availability and reliability. In some customer situations, the workload pressure is more a result of reduced access to the system than a problem with performance.



In a broad sense, the term "availability" can be said to encapsulate the term "reliability." That is, if a system is more reliable, it follows that it should also be more available. But increased availability must also come from a concerted effort to reduce system management time, time spent on tasks which in the past have required that users not be allowed on the system or time during which reduced functionality is in effect. The MPE XL features to be discussed here are the result of a cognizant effort to address our customers' current and future needs for reliable and available systems.

## File System

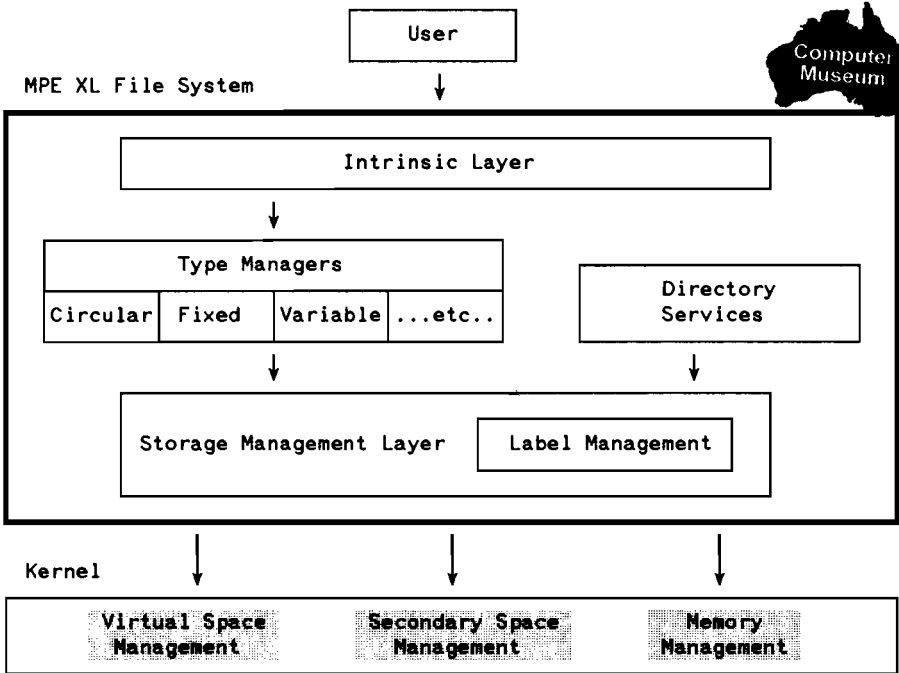
The file system in MPE XL is in many ways evolutionary and is designed with the following objectives:

- Exploit performance advantages of Precision Architecture.
- Compatibility for all non-privileged MPE applications and data.
- Provide a common interface that efficiently supports MPE XL and HP-UX.
- Significantly increase data availability over current MPE. Features will include:
  - 1) Automatic space and file recovery after a soft crash.
  - 2) Optional hard crash recovery in case of disc head crashes.
  - 3) Common services to database and users through transaction intrinsics.
  - 4) Dynamic file backup with users concurrently accessing files.
- Extend file name resolution, accounting, and security features.
- Support future language localization requirements.
- Provide extended features which are desirable to the commercial customer base.
- Provide for data sharing across multiple SPUs.

These objectives will be met in phases as MPE XL matures. Even though essentially 100% compatibility has been provided, the MPE XL file system structure is new. Each important file system function has been partitioned to achieve modularity, extensibility, and leverage of common components. Routines can be straight-forward and designed for single purpose functions. This is in contrast to the MPE file system which is essentially a single path with many special-case branches. The benefits of this modularity should be obvious: better reliability and maintainability.

The MPE XL file system is structured in three layers: 1) the intrinsic layer, 2) the Type Managers layer, and 3) the Storage Management layer. The user application will interface through the intrinsic layer which provides isolation from the details of file access and device management. The actual data is accessed through the Storage Management layer. There will be separate Storage Management modules for discs, tapes, printers, and terminals. Between these two layers reside the Type Managers. There is one Type Manager for each type of file (circular, fixed, variable, etc.); a Type Manager can be added or deleted without affecting any other part of the overall file system.

Other MPE XL components, such as Virtual Space Management (VSM), Secondary Space Management (SSM), Label Management, and Memory Management, play a supporting role to the file system. Figure 1 shows the various layers and modules of the file system.



MPE XL File System (Figure 1)

In the first release, the intrinsic layer will contain both compatibility mode (CM) and native mode (NM) code. Buffered, unbuffered, and multirecord unbuffered access on fixed and variable records will be supported in native mode. Some new intrinsics, such as HPFOPEN, will introduce a more friendly approach to parameter passing. The FOPEN intrinsic, with its multitude of positional parameters, has always been prone to coding errors. In the HPFOPEN implementation, keyword/keyvalue pairs are used, much like the fairly recently introduced FFILEINFO intrinsic. This provides extensibility, flexibility, and a great deal more reliability in coding.

In addition, HPFOPEN is used to gain mapped I/O access to files, a method which provides for greatly improved I/O performance. Mapped I/O eliminates explicit buffering by the file system and provides access to data at the level of LOAD and STORE machine instructions. Logical to physical memory address translation is done in hardware (Translation Lookaside Buffer, or TLB) instead of software. For mapped I/O, HPFOPEN returns a virtual address pointer which is then used by the programmer to access the file. The syntax for HPFOPEN is shown in Figure 2.

```

HPFOPEN(filenum,status[,itemnum,item
 [,itemnum,item
 [,itemnum,item
 .
 .
 [,itemnum,item]...]]]);

```

HPFOPEN Intrinsic Syntax (Figure 2)

## Transaction Management

Transaction Management (XM) is an integral part of the MPE XL file system design. In the current MPE file system, the notion of a *transaction* does not exist. Management of locking, logging, and recovery mechanisms is left to the application, external to the file system. By utilizing operating system services for these mechanisms *inside* the file system, Transaction Management can produce greater efficiency and data integrity. Some externals of XM may not be available to users on the first release of MPE XL.

A transaction can be said to have certain basic properties: 1) *Consistency*, 2) *Atomicity*, and 3) *Durability*. Consistency insures that data is transformed from one state to another in a prescribed manner, i.e. the actions being performed on the data must follow a certain "protocol." Proper serialization of transaction components (reads and writes) in a multi-process, multi-transaction environment is a result of consistency. Atomicity describes the "wholeness" of a transaction; that is, either everything is done or nothing is done. When nothing is done, the transaction is said to be "aborted." When everything is done, it is said to "commit." Durability means that once a transaction is committed, it cannot be annulled; it can survive its creator through disc head crashes and system failures. Transaction Management in MPE XL provides for all of these desirable properties.

The concept of *log sets* is implemented in XM to provide the necessary physical consistency (in case of system crash) and logical consistency (in case of transaction aborts, user process aborts, and system crash). Both before and after images are written to the log so that roll-backward and roll-forward recovery methods are available as appropriate. Before images allow a transaction to be reversed, if necessary, as in a deadlock situation (see below). The recovery mechanism in XM is made to be as automatic as possible. Following a soft crash, it will read the log in order to restore the work of all transactions that did not make it to disc, but which were seen as committed by the user. Any transactions in progress which were not yet committed at the time of the crash will be undone with the AbortTran XM primitive. Hard crash recovery will be roll-forward; starting with a clean, consistent backup of the data, the log file(s) will be used to re-apply all committed transactions.

A log set is defined as a log file and all the files *attached* to the log. The log file itself can be mirrored (that is, a duplicate log file will be maintained) on disc or tape. This is to provide an additional level of protection should the prime log file be destroyed by whatever crash occurs.

Locking within XM can be automatic or it can be controlled by the user. Automatic locking is handled "on-the-fly" as the user accesses virtual pages of data. A set of intrinsics is provided which are used to bound transactions (XMBeginTran, XMEndTran) and determine when locks can be released. Explicit control is given to the user through the XMLockVARange and XMUnlockVARange intrinsics which allow specifying locks on certain virtual address ranges. With any scheme of locking, deadlocks can occur in situations involving multiple processes trying to access the same data. XM will detect deadlocks and resolve them by backing out one of the transactions and thereby releasing its locks. This allows the other transaction to proceed. Notification is given to the application requesting the transaction which was undone so that it can re-try or take some other action.

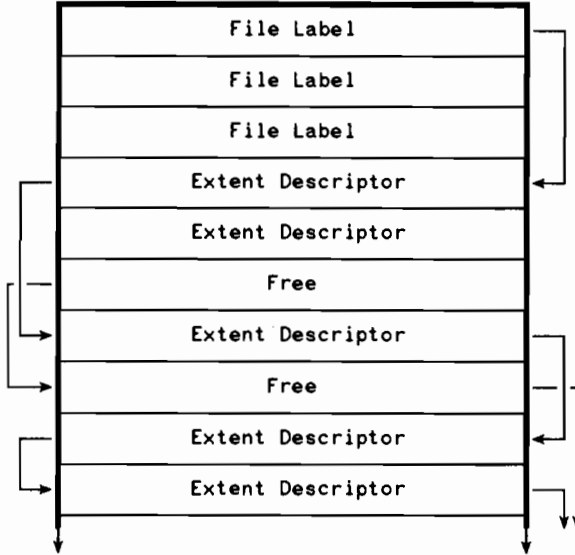
It's not enough to provide just these data integrity enhancements. In a heavy disc transaction environment, *concurrency* must also be maintained to effectively utilize the processing power available. Multiple users on potentially multiple SPUs must be able to gain access to the data at the same time. The concurrency vs. overhead tradeoff problem is addressed in XM by using a unit of lock (virtual address range) and locking strategy which is efficiently supported by Precision Architecture. Lock conflicts are minimized by examining *how* the data is being accessed; a read access for read-only transactions produces a "read shared" lock, allowing concurrent access for other similar transactions.

## File Labels and Extents

In the current MPE file system, the file label is stored on disc immediately in front of the first extent of the file. In MPE XL, file labels and extent descriptors are kept in a special data structure, the **Label Table**, which is separate from the extents themselves. There is one Label Table per disc volume and its location on disc is predefined. The label entries in a particular disc volume's Label Table are for files whose first extents reside on that volume. As in MPE, file extents need not all reside on the same volume (but must be contained within a Volume Set). However, all extent *descriptors* for a given file must reside in the same Label Table.

The Label Table has three types of entries: 1) label entry, 2) extent descriptor entry, and 3) free entry. Each file has one label entry and many (practically unlimited) extent descriptor entries. Related label and extent descriptor entries for a particular file are linked together by standard pointer techniques. Figure 3 shows a simplified view of a Label Table.

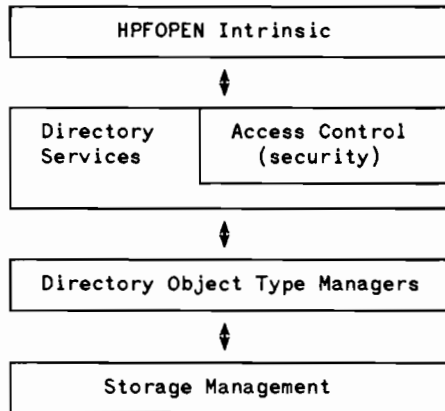
There are many benefits to be had by maintaining file labels and extent descriptors in this way. The Label Table can be very large, limited essentially by available disc space. It follows then that a single file can have practically an unlimited number of extent descriptors and therefore extents. In addition, file system extents in MPE XL are variable in size. The net result is that available disc space is more efficiently utilized, the need for disc condensing is eliminated (increased data availability), and there is no artificial limit on file size.



MPE XL File Label Table (Figure 3)

### System Directory

MPE XL Directory Services resides in the the file system above the Storage Management layer and accesses it through Directory Object Type Managers. Figure 4 expands on Figure 1 and shows the flow of a file open:



MPE XL Directory Services - First Phase (Figure 4)

The Access Control Function (ACF) shown in the preceding diagram is a user definable security mechanism which allows the user to build his own security matrix for a file. In the first phase, ACF will only implement the standard hierarchical security which is available in MPE today (lockword, file level, group level, account level).

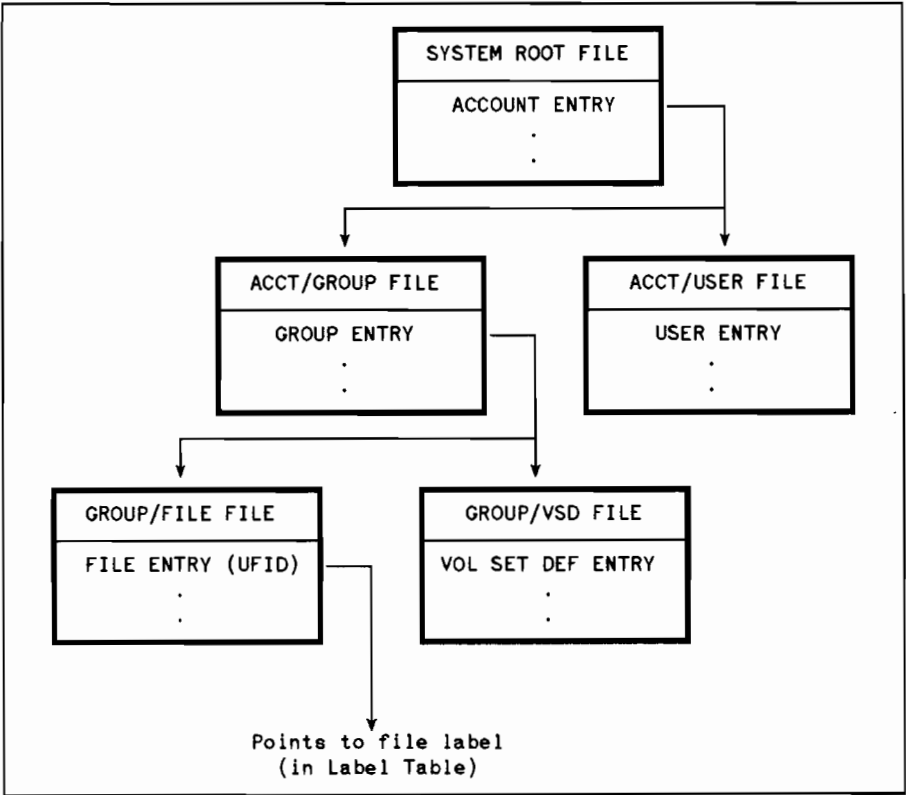
While maintaining complete compatibility with current MPE, the new services and data structures provide for many extensions and eliminate a number of past artificial limitations and hindrances on system availability:

- There is no limitation on the number of files per group, groups per account, users per account, accounts per system, or total directory entries per system.
- Directory "tilting" (which causes difficulties in creating new entries) is eliminated. This problem typically appears in MPE when a large number of files have been incrementally named.
- Directory expansion is automatic; there is no need to re-boot the system from scratch (i.e. RELOAD) in order to reserve more directory disc space.
- The directory SIR (System Internal Resource) is eliminated, greatly improving concurrency.
- The limitation of fixed directory buffering (the directory data segment in MPE) is removed, further improving concurrency. Resolving a file name to a disc address will now be done on the user's stack instead.
- Directory objects (nodes) are spread across discs, resulting in better I/O parallelism and system robustness (potential for continued system operation after a disc failure).
- Provision is made for greater name resolution to support future access methods and additional operating system environments.

The directory is essentially nothing more than a set of files with the ability to grow indefinitely (the practical limit would of course be the available disc space). A diagram of the MPE XL directory structure is shown in Figure 5.

Each file represents a table in the directory and the various tables are connected in tree fashion. As in MPE, the tree is made up of a system root and subtrees. The SYSTEM ROOT file contains account entries. Each account entry points to the GROUP file and the USER file. Each entry in the GROUP file contains a pointer to the FILE file and the VOLUME SET DEFINITION (VSD) file (hang on, it doesn't get much worse!). Each entry in the FILE file maps the file name to a Unique File ID (UFID). The UFID can be viewed as an index into a particular Label Table (see above) where the file label and extent descriptors will be found (whew!). Within each directory file, entries are maintained in alphabetical order.

Since directory objects (tables) are files and may grow to MPE XL limits, the number of users, groups, and accounts which can be supported is practically unlimited. Directory "tilting" is eliminated because there is no logically contiguous space within which expansion is limited. The size of the directory is not artificially limited and can grow automatically; adding entries means simply adding records to files. If a disc goes down, only the file and directory data (if any) on that volume become unavailable. Access to data on other volumes continues as long as there is no need to access the down volume (this is dependent on the level of data partitioning, see Volume Management below). With these improvements in design and implementation, the MPE XL directory will provide increased data availability and system reliability.



MPE XL Directory Structure (Figure 5)

## Volume Management

In MPE XL, Volume Management provides functionality very similar to Private Volumes in MPE. It also introduces a number of new capabilities which will provide increased data availability. MPE *volume sets*, *volume classes*, and *volumes* are supported. A set of operations is provided for the user to manipulate and control these entities as in MPE. These operations should be familiar to the user of Private Volumes and in fact are implemented with similar commands and syntax. A VOLUTIL utility program will replace VINIT and will provide similar but enhanced functionality.

In MPE, there are two distinct disc *domains*: system and non-system. Private volumes and serial volumes are examples of discs which reside in the non-system domain. In a broad sense, MPE XL makes "system" volumes more like "private" volumes and "private" volumes a little like "system" volumes. Thus in MPE XL we have only one disc domain. Within this domain are volume sets. One of the volume sets is defined as the "system" volume set. (Currently the system volume set has the formal name HPE\_SYSTEM\_VOLUME\_SET). A volume is defined to be a disc pack and is self contained on the media. This means that the volume set *definition* moves with the volume set, unlike MPE where the system volumes are defined by device configuration.


In the past, there has been some reluctance on the part of users to implement Private Volumes. It is hoped that MPE XL users will make substantial use of Volume Management capabilities, since the essential benefit is that the system will be more robust in the event of disc failures and related crashes. When data storage is partitioned, control and protection of the data is improved. Volume Management will initially support three levels of data partitioning: 1) Volume Set, 2) Volume Class, and 3) Volume. The volume level is the most granular and offers the greatest protection and control. Although the user must be involved in determining granularity, Volume Management simplifies the task as much as possible and provides benefits which are well worth the effort.

For an excellent expanded discussion of MPE XL Volume Management, see "HPE Volume Management", by Rick Ehrhart, as published in the proceedings of the Detroit INTEREX conference (September, 1986).

## MPE XL Backup/Recovery

Perhaps one of the most frustrating day-to-day experiences for HP3000 users is to be asked to "get off the system" during the backup. Beyond the frustration is the very real fact that backup time can have a tremendous impact on system availability. For all practical purposes, backup time is system down time in MPE. In fact, research has shown that out of whatever "down" time our average Series 68 customer experiences, 69% of it is for backups.



	HP 3000	<b>RI02/10</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

A related problem is storage capacity and the saturation of system resources which can result when backing up large amounts of online data. With the new Precision Architecture systems, disc storage capacities will be increasing to meet the needs of our customers' applications. New peripheral backup devices will become available, with large bandwidth capabilities for transferring data. All of these factors together made it clear that MPE STORE/RESTORE needed improvements. In MPE XL, native STORE/RESTORE has these objectives:

- 100% availability of data during the backup (STORE).
- Improved performance, efficiently utilizing Precision Architecture and future peripherals.
- Not greater than 50% CPU and I/O bandwidth utilization during the backup.
- Easy to use with minimum operator intervention, especially for "office environment" configurations.
- Support for backup of data distributed within a network.
- Provide backward and forward file transport capability between MPE and MPE XL systems.
- Provide support and proper security for inclusion of the system directory on STORE tapes.
- Provide a supported method for tape verification (VSTORE utility).
- Extensibility for new features and technologies.

As in other parts of MPE XL, a phased approach will be taken in meeting these objectives. Native STORE/RESTORE will utilize a number of techniques to meet the above objectives and provide the necessary backup/recovery services, including:

- **Static backup.** This is the current implementation in MPE; files are inaccessible during backup.
- **Dynamic backup.** Files are fully accessible for all types of access (read, write, etc.) during the backup. Modifications made to files being STOREd are logged; the log files are saved on the backup medium along with the fileset being STOREd. On RESTORE, both the data and the log file are used to recover the data to a consistent state. For complete consistency of all file types, the system should be quiescent for a short period of time when the backup is started. Dynamic backup is implemented with the file system's Transaction Management (XM) services.
- **Transport backup.** STORE tapes are created in MPE format; only those files which do not exceed standard MPE limitations can be written to the tape (i.e. a file with more than 32 extents cannot be STOREd with the transport method).
- **Interleaving.** Multiple files on different disc drives are read concurrently and interleaved in the data stream to the backup device, typically on extent boundaries. The intent is to maintain "streaming mode" or maximum throughput on the backup peripheral.

- **Consecutive backup devices.** A *tapeset* is defined with multiple tape drives. When one device begins rewind, the next device in the tapeset will immediately begin writing. Thus rewind and reel switch time can overlap with tape writing, increasing throughput. See Figure 6 for a sample scenario.
- **Concurrent backup devices.** Multiple, concurrently writing devices. This increases the number of parallel paths to the backup media (depending on I/O system configuration limitations).

	STEP 1	STEP 2	STEP 3	STEP 4	STEP 5	STEP 6
TAPE DRIVE 1	WRITE →	REWIND ←	WAIT (change tape)	WRITE →		REWIND ←
TAPE DRIVE 2	WAIT (tape ready)	WRITE →		REWIND ←	WAIT (change tape)	WRITE →

Consecutive Backup Devices (Figure 6)

When all of the new features are implemented, the new syntax for STORE will appear as shown in Figure 7. The new options should be readily apparent.

```


:STORE [filesetlist] [;storefile] [;option[;...]]

 where option is
 [;SHOW[=showparmlist]]
 [;ONERROR=recoverytype]
 [;FILES=maxfiles][;DATE<=accdate]
 [;DATE>=moddate]

 [;PURGE]
 [;PROGRESS [#minutes]]
 [;STORESET=(device[,...]),(device[,...]),...]
 [;INTER]
 [;DYNAMIC]
 [;DIRECTORY]
 [;LOGONLY]
 [;TRANSPORT]

```

STORE Command Syntax (Figure 7)

	HP 3000	<i>R102/12</i>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

In the first release of MPE XL, the DYNAMIC, LOGONLY, and STORESET options will not be available. Default backup will be static unless the DYNAMIC option is specified. The LOGONLY option specifies that only the transaction log files from any log sets are to be stored. The TRANSPORT option is mutually exclusive with the DYNAMIC, LOGONLY, STORESET, INTER, or DIRECTORY options. INTER specifies that file interleaving is to be used. The STORESET option, which is used instead of <storefile>, is the most interesting. STORESET is used to specify consecutive backup, concurrent backup, or both. Consecutive tapes are specified in the following way:

```
;STORESET = (*tape1,*tape2,*tape3)
```

STORE will select the first drive it finds in a ready state. Concurrent devices are specified by:

```
;STORESET = (*tape1),(*tape2),(*tape3)
```

In this example, all three tapes will be used in parallel. Concurrently accessed consecutive tape sets would be specified by:


```
;STORESET = (*tape1,*tape2),(*tape3,*tape4)
```

In this example, two tapes would be storing at any given time while the other two rewind and change reels.

It should be clear that with these new features, native STORE/RESTORE will provide substantial improvements in system availability and backup scheduling flexibility (reduced need for extra shifts just to do backups). In conjunction with intelligent partitioning of data using the capabilities of Volume Management, and with the appropriate number of backup devices, native STORE/RESTORE will also provide enhanced performance.

## Summary

The list of enhancements and features in MPE XL which has been examined here is by no means exhaustive. Other contributions to system availability, such as online device configuration (for printers, tapes, and terminals) and system tables which self-configure and automatically adjust in size if necessary, are continually being evaluated and implemented in MPE XL. New methods of system analysis and design are being incorporated to insure quality in the code produced. The net result of all of this is simple: a computer system which has the necessary features, performance, and reliability to support our customers' growing application and workload requirements.

	HP 3000	<i>R103/1</i>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## DESIGN FEATURES OF THE MPE XL USER INTERFACE

by *Jeff Vance, John Korondy, & Denis Rachal*  
*Hewlett-Packard*

### ABSTRACT

The MPE XL User Interface, consisting of commands, expressions, variables and intrinsics, encapsulates the majority of communication between the end user and the machine. This new User Interface was designed to satisfy several important goals. First, and foremost, it had to be compatible with MPE V/E. Second, it had to furnish the foundation technology for localizing the User Interface. Third, it had to provide a powerful, flexible, and productive environment for the general user as well as the very experienced user. Fourth, it was desirable to expose some of the architectural advantages of the HP Precision Architecture.

This paper describes the new MPE XL User Interface, focusing on the extensions beyond MPE V/E. The command language aspect of the User Interface is emphasized, and examples are provided to illustrate how we've achieved the four primary objectives mentioned above. Knowledge of the MPE V/E User Interface and the basic use of the new MPE XL commands is assumed.

#### I. MPE V/E COMPATIBILITY - CM vs. NM commands, parsing, error management.

Most of the MPE V/E commands (e.g. run, listeq, fcopy, showjob) have been "ported" to MPE XL without modification. These commands are typically executed in compatibility mode (CM), retain their original syntax (parsing) and their original error/warning conditions. Some MPE V/E commands do their parsing in CM and then switch to native mode (NM) to complete their execution. The listf, listacct, and report commands have been implemented this way, so that the external interface is unchanged despite the fact that internal

interfaces are considerably different. And, finally, some MPE V/E commands, and all new MPE XL commands, execute in NM. Examples include: redo, setcatalog, copy, chgroup and print. All MPE V/E commands which have been enhanced (e.g. redo) will continue to function as in MPE V/E. The new features can only be invoked by supplying additional parameters.

## II. CONSISTENCY AND LOCALIZATION - central parser, prompt strings.

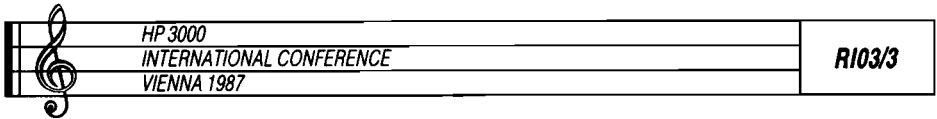
All native mode (NM) commands are parsed by a centralized parser, which enforces a consistent syntax and provides the ability to localize command input. Localization of command output can be done via the message catalogs.

Each NM command has a parse template, called a "prompt string", which is used by the parser to recognize the entire syntax for the command. This includes the spelling and position of every required and optional command parameter. Prompt strings can be generated in many languages, thus the command could be parsed and executed regardless of the input language.

Localization is not available on first release because many MPE XL commands are still parsed and executed in compatibility mode; and also many commands still produce hard-coded output. However, the foundation is in place for the entire User Interface to be localized.

## III. EVOLUTION OF THE COMMAND INTERPRETER (CI) - command files, search path, variables, expressions.

The majority of MPE XL User Interface enhancements are aimed to make the Command Interpreter (CI) more powerful, more flexible, and easier to use than its predecessor. This can be viewed as the evolution of the CI from a command interpreter to an interpretive command language. This evolution is necessary to support the diverse and complex requirements of the non-programmer end-user, as well as addressing the needs of the experienced user and programmer. Like most programming languages, the CI supports variables, expression evaluation, and logic flow control via recursion, iteration (while) and conditional constructs (if, else, endif).



This section contains five parts. First, command files are discussed and compared to UDCs. Next, the CI's search path for command files and program files is described. Part C explains CI variables as a powerful extension of JCWs. Next, expression evaluation is described, and lastly, several command file examples are provided which tie together variables and expressions.

#### A. Command Files.

Command files are simply files which contain one or more commands. The command lines are referred to as the "body" of the command file. A command within a command file may be an MPE XL command, a UDC command, or another file name. A command file may contain a header, which defines parameters and options (same as UDC options). As with UDCs, command files cannot contain data following the command line. If a command file accepts parameters the first record must begin with "PARM" followed by the parameter list. For example: PARM p1,p2=10,p3="". This parameter line indicates that there are at most three parameters, the first parameter is required, the second parameter is optional with a default value of 10, and the last parameter is also optional with a default value of "" (nil).

#### Command Files vs. UDCs -

Command files are very similar to UDCs in that they create customizable commands. There are three major differences between command files and UDCs:

First, since UDCs are searched before MPE XL commands, they can effectively override or redefine an existing MPE XL command. Command files are looked for after MPE XL commands, and thus cannot be used for this purpose.

Second, UDCs are cataloged and command files don't have this requirement. This has 3 implications: a) new UDCs cannot be created and old UDCs cannot be modified without first un-cataloging the UDC file and then re-cataloging it; whereas, command files don't have this constraint. b) a UDC file is only opened once, at logon time, but a command file must be opened and closed for each invocation.

c) UDCs provide more efficient organization for command scripts, and use less disc space than if each UDC command were kept as a separate command file.

Third, UDCs can control whether or not recursion is permitted via OPTION RECURSION. Command files do not support the control of recursion (although recursion is legal).

In summary, UDCs should be chosen for frequently used, stable commands, and for command name aliasing. For performance reasons, recursion should be confined to UDCs. Command files are best suited for scripts that may be enhanced often, for testing of potential UDCs, and for environments where dynamic search paths are exploited. HELP is available for UDCs, command files, and program files.

#### B. Search Path and Implied Run.

The CI follows a user-definable search path when looking for command files programs. This path is only examined if the user's command is not a UDC and is not a built-in (known) MPE XL command. The HPPATH predefined variable contains the current search path. If HPPATH is null (:setvar hppath "") then no command files will ever be found. The default value for HPPATH is "!HPGROUP, pub, pub.sys". If the command entered is X (assume X is not a UDC) then the file X.logon group (!HPGROUP) will be searched for. If it is not found then X.pub is looked for. If, again, it is not located then X.pub.sys is examined. If, after this final attempt, the file cannot be found then the command X is assumed to be an unknown command name and CIERR 975 is reported.

On the other hand, if a file named X is located in one of the groups specified by HPPATH then its file code is read to determine if the file is a program file (file code of 1029 or 1030), a command file (file code within zero through 1023), or neither. The search path can be altered via the setvar command. To illustrate, if most of your command files reside in a group named "scripts", in your logon account, then changing HPPATH can speed up command file searching. For example, :SETVAR HPPATH 'scripts, '+ HPPATH would add

the group "scripts" to the current search path. The SETVAR command, variables, and "!" dereferencing are discussed in part C.

#### Implied Run -

Program files use the same search path as command files. Using the default path, SPOOK.PUB.SYS can be invoked simply by issuing "spook". This is referred to as "implied run". Implied run commands have two optional parameters. The first parameter is assumed to be an info string, and the second parameter is interpreted as the parm= RUN command parameter. Other options and keywords on the RUN command (debug, lmap, unsat=, stdlist=, etc.) are not supported by the implied run facility.

#### C. Variables.

Commands: setvar, showvar, deletevar, setjcw, showjcw.

Intrinsics: hpciputvar, hpcigetvar, hpcideletevar, setjcw, getjcw, putjcw, findjcw.

An MPE XL variable is similar to an MPE V/E jcw. However variables can contain 32 bit integers, string values, boolean TRUE or FALSE, or the name of other variables (also a string value). Like jcw, variables are job/session global meaning that any process within a given job or session can read (and possibly modify) a variable. There are two major classes of variables: user-defined via the new SETVAR command or the HPCIPUTVAR intrinsic, and HP predefined variables such as HPACCOUNT, HPDATEF, etc. The new MPE XL Commands Manual contains a list of all predefined variables. All user-defined variables can be read, modified and deleted. Most HP predefined variables are read-only, however several important variables can also be modified (for example: HPPROMPT, HPPATH, HPCMDTRACE, HPAUTOCONT, HPMSGFENCE, HPREDOSIZE, HPRESULT, HPTIMEOUT). None of the predefined variables may be deleted. Four variable types are supported: integer, string, boolean and jcw. If a variable is created via the setjcw command then it becomes a jcw type variable and a warning is issued if it is changed to a value not supported by MPE V/E jcw.



The value of a variable is retrieved by placing an exclamation point (!) before the variable name, e.g. !MY\_VAR. This is called dereferencing the variable: the name "MY\_VAR" is replaced by its value. For example, if MY\_VAR contains the string 'hi there' (e.g. setvar my\_var 'hi there') then every occurrence of !MY\_VAR is substituted by the string "hi there". This is the same substitution that is done for UDC parameters. A variable's value may also be retrieved when its name is used within an expression (this is implicit dereferencing, whereas !varname is explicit dereferencing). Two examples of implicit dereferencing follow:

```
:while (i < len(Var_x))
: setvar my_calc hresult+10
```

As mention above, a variable may contain the name of another variable. This is essential for read/write variables whose values are dynamic. For example, if you want to personalize the CI's prompt to contain your username followed by "(current command number):", e.g. JEFF(10):, then, since the command number is always changing, a dynamic variable is required. This prompt can be obtained by the following command:

```
:setvar hpprompt '!hpuser(!!hpcmdnum):'
```

The quotes cause the variable's type to be string. The first "!" dereferences the value of the HPUSER predefined variable (i.e. "JEFF"). The next two exclamation points ("!!") fold to a single exclamation point followed by the string "hpcmdnum" (also a predefined variable). Thus, a showvar of HPPROMPT would reveal "JEFF(!HPCMDNUM):" as its value. To see how this translates to "JEFF(10):" recursive dereferencing needs to be explained. Whenever a variable is explicitly dereferenced (!varname) substitution is done until all dereferences within varname's value have been attempted. The following simple example will illustrate this:

```
:setvar a "!!b" {!b is stored as the value of a}
:setvar b 10 {10 is stored as the value of b}
:echo Here is A: !a {"Here is A: 10" will be echoed}
```

!a is dereferenced as !b, which in turn is dereferenced as 10. SHOWVAR always retrieves the immediate value of a variable. This is also true for implicit dereferencing. However, explicit dereferenc-

ing always fully substitutes the variable. So, in the HPPROMPT example above, since the prompt contains an exclamation point, indicating another variable needs to be retrieved, HPPROMPT will be fully dereferenced before it is displayed. Thus the final prompt would be "JEFF(10):" when the current command number is ten, and will always track the command number.

QUESTION - what would the prompt be if a single explanation point was used rather than the two, e.g. setvar hpprompt '!hpuser(!hpcmdnum):'?

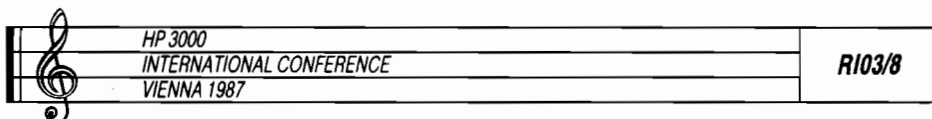
(Answer: the prompt would contain the current command number as a static variable. For example if the command number was 12, the prompt would always show 12, regardless of the increasing command number.)

#### D. Expressions.

Commands: calc, setvar, if, while.

Intrinsics: hpcicommand, command.

Expressions are permitted in five commands: calc, setvar, if, while and setjcw. The following discussion does not pertain to setjcw, which retains MPE V/E expressions. The expression evaluator supports a rich set of functions. The entire list is available in the MPE XL Commands Manual. All arithmetic operations are allowed, including: absolute value (ABS), modulo (MOD) and exponentiation (^). Many string functions (argument is a string) are defined, such as: concatenation (+), length (LEN), ordinal (ORD), extraction (LFT, RHT, POS, STR), and case shifting (DWNS, UPS). Special variable functions include: existence (BOUND) and type (TYPEOF). Bit operations supported include: bitwise and (BAND), bitwise or (BOR), bitwise not (BNOT), bitwise exclusive or (BXOR), shift left (LSL), and shift right (LSR). Numeric conversion functions are: convert to octal (OCTAL) and convert to hexadecimal (HEX). Finally, some special file functions provided via FINFO include: file existence (FINFO(0)), file creation date (FINFO(6)), file modification date (FINFO(8)), file code (FINFO(9)), foptions (FINFO(13)) and others.



Mixed expressions are not allowed, for example, calc "a"+1 would result in an error. Standard precedence rules apply, with variable dereferencing superseding all other operations.

Below are some examples using variables and expression evaluation:

```

:setvar a 'aa' {a = aa}
:setvar b 'BB'+a {b = BBaa}
:setvar c (len(b)+pos("a",b)) lsl 1 {c = (4+3) lsl 1=14}
:if bound(a) and ord(ups(a))=65 then {T AND T = TRUE}
:calc finfo(b,0) or ups(str(b,2,1))="X" {T(file "BBAA" exists)}
 {OF F=TRUE }

```

#### E. Command File Examples.

The first command file is more fully commented than the others and should be read thoroughly.

1. "CIERR" - prints the error message text associated with either the current value of the CIERROR jcw, or a passed cierror value.

```

PARAM cierr=0
OPTION nolist
COMMENT Parameter cierr is not required. Its default value is
COMMENT 0. The "parm" and "option" lines comprise the header.
COMMENT
COMMENT If cierr is not zero then save the current cierror jcw
COMMENT and set the cierror jcw to cierr.
COMMENT Note that parameters must always be explicitly
COMMENT dereferenced. Implicit dereferencing is only available
COMMENT to global variables.
COMMENT
if !cierr <> 0 then
 COMMENT Since we're dealing with global variables,
 COMMENT use an unlikely name.
 COMMENT
 setvar _cierr cierror
 setjcw cierror !cierr
endif
COMMENT Retrieve the error text associated to the value of

```

```

COMMENT CIERROR.
COMMENT
setvar _text hpcierrmsg
if len(_text) = 0 then
 comment No error message corresponding to CIERROR.
 setvar _text 'INVALID CI ERROR NUMBER (!cierror).'
endif
COMMENT Display the final message to $stdlist.
echo !_text
COMMENT Cleanup by deleting global variables used here and
COMMENT resetting CIERROR if necessary.
COMMENT
deletevar _text
if !cierr <> 0 then
 setjcw cierror !_cierr
 deletevar _cierr
endif
COMMENT End of command file.

```

Usage:

```

:setjcw cierror 9072
:cierr 9103
THIS COMMAND IS NO LONGER SUPPORTED IN MPE XL. (CIERR 9103)
:cierr 2
INVALID CI ERROR NUMBER (2).
:cierr
THERE ARE NO COMMANDS AVAILABLE TO REDO. (CIERR 9072)

```

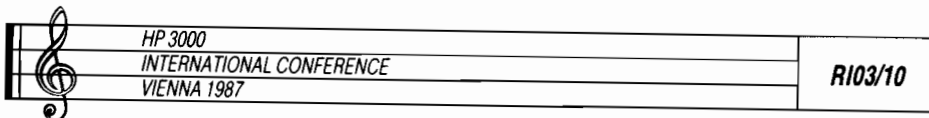
This command file can be improved by adding parameter verification code, i.e., test if the parameter is fully numeric. For example:

```

if typeof(!cierr) <> 1 then
 comment 0=bad expression, 1=numeric, 2=string, 3=boolean.
 echo Expected optional parameter to be numeric.
else . . .

```

2. "TAIL" - this simple command file prints the last n records from a given file.



```
PARM file, last=10
COMMENT Print the last "last" records from "file".
print !file;start=-!last
COMMENT End of include file.
```

Usage:

```
:tail catalog.pub.sys
:tail myletter 40
```

3. "ME" - this command file provides a convenient :showme format.

```
PARM flag=''
COMMENT Shows user environment info similar to :showme. If
COMMENT "flag" is non-null then the user's capabilities are
COMMENT also displayed.
COMMENT
setvar _me "!hpuser.!hpaccount,!hpgroup (Ldev=!hpldevin)&
#!hpjobtype!hpjobnum"
if len(hpjobname) > 0 then
 setvar _me "!hpjobname,"+_me
endif
if hpinbreak then
 setvar _me _me+ " <break>"
endif
echo !_me
if "!flag" <> "" then
 echo !hpusercapf
endif
deletevar _me
COMMENT End of command file.
```

Usage:

```
:me
JEFF.UI,CI (Ldev=21) #S12
:me x
MYID,MANAGER.SYS,SCRIPTS (Ldev=20) #S11 <break>
AM,AL,GL,DI,CV,UV,LG,CS,ND,SF,IA,BA,PH,DS,MR,PM
```

4. "ADDCAP" - this command file adds one to nine more capabilities to a given user. The default user is the current user name.

```

PARM user='',cap,c2='',c3='',c4='',c5='',c6='',c7='',c8='',c9=''
COMMENT Adds cap (required) thru c9 to the "user"s current
COMMENT caps. Prompts for re-logon.
COMMENT
setvar _captemp ups("!hpusercapf,!cap,!c2,!c3,!c4,!c5,!c6,!c7,&
!c8,!c9")
COMMENT Strip trailing commas.
trim _captemp ", "
if 'user' = '' then
 setvar _user hpuser
else
 setvar _user ups('user')
endif
setjcw cierror 0
continue
altuser !_user;cap=!_captemp
if (cierror <> 0 and (cierror <> 750) then
 echo (ADDCAP): !_user's capabilities remain unchanged: &
!hpusercapf.
else
 echo (ADDCAP): !_user's capabilities are: !_captemp.
 comment If "user" is current user then prompt for re-logon.
 if hpjobtype = 'S' then
 if '!_user' = hpuser then
 setvar _captemp 'n'
 input _captemp, "(ADDCAP): Logoff now to obtain new &
capabilities?";wait=10
 if (cierror <> 0) or (dwms(lft(_captemp,1))<>'y') then
 echo (ADDCAP): New capabilities take effect at next&
logon
 else
 hello !hpjobname,!hpuser.!hpaccount,!hpgroup;&
info='echo (ADDCAP): Done.'
 endif
 else
 echo (ADDCAP): New capabilities take effect next time &
!_user on.

```

```

endif
endif
endif
deletevar _captemp, _user
COMMENT End of command file.

```

Usage: (assume logon user's (MGR) capabilities are SM,AM,BA,IA)

```

:adddcap ,ph,al
(ADDCAP): MGR's capabilities are: SM,AM,AL,BA,IA,PH.
(ADDCAP): Logoff now to obtain new capabilities?
no
(ADDCAP): New capabilities take effect at next logon.

:adddcap cap=op
(ADDCAP): MGR's capabilities are now: SM,AM,AL,OP,BA,IA,PH.
(ADDCAP): Logoff now to obtain new capabilities?
yes
<new session banner>
(ADDCAP): Done.

:adddcap accting,cv uv al
(ADDCAP): ACCTING's capabilities are now:
SM,AM,AL,OP,CV,UV,BA,IA,PH.
(ADDCAP): New capabilities take effect next time ACCTING logs on.

```

5. "TRIM" - this command file trims leading or trailing characters from a supplied variable.

```

PARAM varname, trimchar=" ", from=RIGHT
COMMENT Trims all "trimchar" from "varname" starting at the
COMMENT "from" side.
COMMENT
if not(bound(!varname)) then
 echo (TRIM): The variable !varname is not defined.
else
 if ups(lft('!from',1)) <> "L" then
 comment Trimming from the right of varname.
 setvar _trimside 'RHT'
 setvar _trimsave 'LFT'
 else

```



```
 setvar _trimside 'LFT'
 setvar _trimsave 'RHT'
 endif
while (len(!varname)>0) and (!_trimside(!varname,1)="!trimchar")
 setvar !varname !_trimsave(!varname,len(!varname)-1)
endwhile
deletevar _trimsave, _trimside
endif
COMMENT End of command file.
```

Usage:

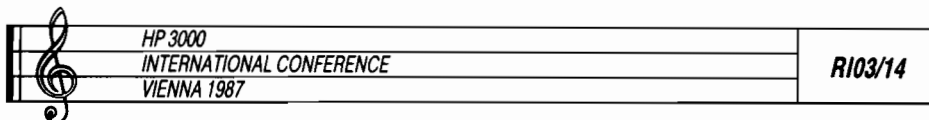
```
:setvar foo "abcd" :setvar foo '...abcd'
:trim foo :trim foo '.' LEFT
:echo !foo :echo !foo
abcd abcd

:setvar foo " abc" :setvar foo 'abcdddd'
:trim foo :trim foo 'd'
:trim foo;from=left :echo !foo
:echo !foo abc
abc
```

6. "RSIZE" - this command file does one of three things depending on the optional parameter. If the parameter is omitted then the current size of the redo/history stack is displayed. If the parameter is signed, e.g. +10, then the redo stack size is adjusted relative to its current size. If the parameter is unsigned then the redo stack size is set to the parameter value.

```
PARAM size=""
COMMENT 1) echoes current size of redo stack, or 2) adjusts
COMMENT hpreddosize to hpreddosize+"size", or 3) sets hpreddosize
COMMENT to "size".
COMMENT
if "!size" = "" then
 echo (RSIZE): The redo stack size is !hpreddosize.
else
 if typeof(!size) <> 1 then
 comment Not numeric.
 echo (RSIZE): If parameter is supplied it must be a &
```





signed or unsigned number.

```

else
 if (lft("!size",1) = "+") or (lft("!size",1) = "-") then
 setvar _size !hpredosize+!size
 setvar _signed true
 else
 setvar _size !size
 setvar _signed false
 endif
setjcw cierror 0
continue
setvar hpredosize _size
if cierror <> 0 then
 echo (RSIZE): The redo stack size remains !hpredosize.
else
 if _signed then
 echo (RSIZE): The redo stack size is now !hpredosize.
 endif
endif
deletevar _signed, _size
endif
endif
COMMENT End of command file.

```


Usage: (assume the current redo stack size is 20)

```

:rsize
(RSIZE): The redo stack size is 20.
:rsize +15
(RSIZE): The redo stack size is now 35.
:rsize 50
:rsize -10
(RSIZE): The redo stack size is now 40.

```

7. "LOGPUR" - this command file purges all log#### log files in pub.sys. To better performance the user may specify the starting log file number. The purged log file name will be echoed if the second parameter is not "QUIET".

	HP 3000	<b>RI03/15</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

```

PARM lognum=0,mode='QUIET'
COMMENT Purge all log####.pub.sys starting at "lognum". Echo
COMMENT purged file if not "quiet".
COMMENT
setvar _logcnt 0
setvar _logx !lognum
setvar _quiet ups(lft(!mode,l)) = 'Q'
setvar _cont hpautocont
setvar _msg hpmsgfence
setvar hpautocont true
setvar hpmsgfence 2
setjcw cierror 0
COMMENT Find first log file to purge.
setvar _logname "LOG"+str('0000',l,4-len(!_logx))+"!_logx"
purge !_logname
while cierror = 383 or cierror = 0
 if cierror = 0 then
 setvar _logcnt _logcnt+1
 if not (_quiet) then
 echo (LOGPUR): !_logname.PUB.SYS has been purged.
 endif
 endif
 setvar _logx _logx+1
 setvar _logname "LOG"+str('0000',l,4-len(!_logx))+"!_logx"
 setjcw cierror 0
 purge !_logname
endwhile
setvar hpautocont _cont
setvar hpmsgfence _msg
echo (LOGPUR): !_logcnt log files were purged.
deletevar _logcnt, _logname, _logx, _quiet, _cont, _msg
COMMENT End of command file.

```

Usage: (assume LOG0030, LOG0031, LOG0032, LOG0033, LOG0034#)

```

:logpur
(LOGPUR): 4 log files were purged.
:logpur 30
(LOGPUR): 4 log files were purged.
:logpur 32,loud
(LOGPUR): LOG0032.PUB.SYS has been purged.

```

(LOGPUR): LOG0033.PUB.SYS has been purged.

(LOGPUR): 2 log files were purged.

8. "FINFO" - displays file label information for a given filename.

PARM file

COMMENT Use finfo to show file label info.

COMMENT

```

if not(finfo('!file',0)) then
 comment File does not exist.
 setvar _file ups('!file')
 if lft('!file',1) <> '#' and lft('!file',1) <> '$' then
 comment Qualify file before reporting non-existence.
 setvar _pos pos('.',_file)
 if _pos = 0 then
 setvar _file _file+'.!hpgroup.!hpaccount'
 else
 if pos('.',rht(_file,len(_file)-_pos)) = 0 then
 setvar _file _file+'.!hpaccount'
 endif
 endif
 deletevar _pos
 endif
 echo (FINFO): !_file does not exist.
 deletevar _file
else
 comment ** formal file designator **
 setvar _finfo finfo('!file',1)
 echo (FINFO): Full file description for !_finfo follows:
 echo
 comment ** creator and create/modify dates **
 setvar _finfo finfo('!file',4)
 setvar _finfo1 finfo('!file',6)
 setvar _finfo2 finfo('!file',8)
 setvar _finfo3 finfo('!file',24)
 echo Created by !_finfo on !_finfo1.
 echo Modified on !_finfo2 at !_finfo3.
 comment ** file code, rec size, eof, flimit **
 setvar _finfo finfo('!file',9)
 setvar _finfo1 finfo('!file',-9)

```

```

if len(_finfo) = 0 then
 setvar _finfo '!_finfol'
else
 setvar _finfo _finfo+' (!_finfol)'
endif
setvar _finfol finfo('!file',14)
setvar _finfo2 finfo('!file',19)
setvar _finfo3 finfo('!file',12)
echo Fcode: !_finfo, Recsize: !_finfol, Eof: !_finfo2, &
Flimit: !_finfo3.
comment ** foption **
setvar _finfo finfo('!file',13)
setvar _finfol finfo('!file',-13)
setvar _finfo2 octal(_finfol)
setvar _finfo3 hex(_finfol)
echo Foption: !_finfo (#!_finfol, !_finfo2, !_finfo3).
deletevar _finfo, _finfol, _finfo2, _finfo3
endif
COMMENT End of command file.

```

Usage:

```
:finfo sl
```

(FINFO): Full file description for SL.PUB.SYS follows:

Created by MANAGER on TUE, DEC 9, 1986.

Modified on TUE, DEC 9, 1986 at 7:00 PM.

Fcode: SL (1031), Recsize: -256, Eof=3919, Flimit=10000.

Foption: BINARY, FIXED, NOCCTL, STD (#1025, %2001, \$401).

```
:build a;msg;rec=-80,,f,ascii;cctl
```

```
:finfo a
```

(FINFO): Full file description for A.PUB.SYS follows:

Created by JEFF on WED, OCT 22, 1986.


Modified on FRI, OCT 24, 1986 at 6:45 AM.

Fcode: 0, Recsize: -81, Eof=0, Flimit=1029.

Foption: ASCII, VARIABLE, CCTL, MSG (#12613, %30505, \$3145)

```
:build temp;temp;rec=40,,f,ascii;disc=100
```

```
:file t=temp
```

	HP 3000	<i>R103/18</i>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

:finfo \*t

(FINFO): Full file description for TEMP.PUB.SYS follows:

Created by MGR on FRI, OCT 24, 1986.

Modified on WED, DEC 10, 1986 at 2:26 PM.

Fcode: 0, Recsize: -80, Eof: 0, Flimit: 100.

Foption: ASCII, FIXED, NOCCTL, STD (#1020, %2006, \$406).

:build \$newpass

:finfo \$oldpass


(FINFO): Full file description for \$OLDPASS.PUB.SYS follows:

Created by JEFF on SAT, OCT 25, 1986.

Modified on SAT, OCT 25, 1986 at 11:30 AM.

Fcode: 0, Recsize: -256, Eof: 0, Flimit: 1023.

Foption: BINARY, FIXED, NOCCTL, STD (#1050, %2032, \$41A).

	HP 3000	<b>R103/19</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## BIOGRAPHICAL SKETCH

Name: *Jeff Vance*

Title: *Member of the Technical Staff*

Employer: *Hewlett-Packard Information Software Operation*

Job Description: *Design/Implementation of the new MPE XL Command Interpreter.*

Background: *Joined Hewlett-Packard in 1979 and worked four years in MIS applications for Inventory Control and MRP systems. He has held his current position for three years.*

Education: *BS, University of California at Davis, 1979*

Name: *John Korondy*

Title: *Project Manager, User Interfaces, Operating Systems Laboratory*

Employer: *Hewlett-Packard Information Software Operation*

Job Description: *Responsible for the design, development, and integration of User Interface Software of the MPE XL Operating system.*

Background: *Joined Hewlett-Packard in 1979 and spent over three years in Information Systems development and management. In the past four years, John has contributed to the design and development of the MPE XL Operating System, in the implementation of the Low-Level I/O software subsystem, and more recently, the User Interface software. He has been a project manager for three years.*

Education: *BSCS Magna cum Laude, University of California at Los Angeles, 1979*

Name: *Denis Rachal*

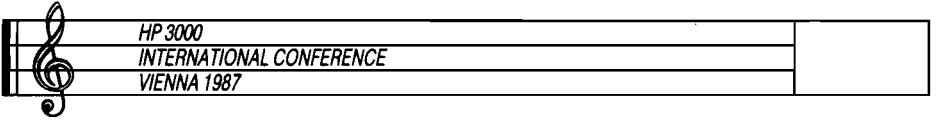
Title: *Member of Technical Staff*

Employer: *Hewlett-Packard Information Technology Group*


Job Description: *Provide support for the MPE XL operating system through internals course development, and problem analysis.*

Background: *Joined Hewlett-Packard in 1979 and worked as a Customer Engineer for one and a half years in the Neely Santa Clara Office. He was then promoted and worked as an HP3000 Technical Support Engineer for the next five years. Denis has held his current position at ITG for one and a half years.*

Education: *BSEE, University of Southern California, 1979*



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

	<i>HP 3000</i>	<i>R104/1</i>
	<i>INTERNATIONAL CONFERENCE</i>	
	<i>VIENNA 1987</i>	

**MPE V to MPE XL Migration Overview**

**by  
R. Gregory Stephens**

**Hewlett-Packard Company.  
19447 Pruneridge Ave.  
Cupertino, CA 95014**



## Abstract

This paper provides an overview of the Migration Process from MPE V based HP 3000 systems to MPE XL based HP 3000 systems. Topics addressed include Education, Predelivery Planning, Preparation, System Installation, as well as 900 Series Compatibility Mode and Native Mode Execution. The role of Migration Tools and Services in this process is also discussed.

## Introduction

Migration to 900 Series HP 3000 systems is based on full compatibility with the rest of the HP 3000 family. This compatible migration path protects your investment in HP 3000 hardware and software and provides a painless growth path to the high performance and capabilities of the 900 Series systems.


Hewlett-Packard has made a significant investment in providing a high degree of compatibility and ease of migration. This investment is exemplified in the areas of Object Code Compatibility, Source Code Compatibility, Database Compatibility, Network Compatibility, Operational Compatibility, as well as the Migration Tools and Services that are being offered.

This paper presents an overview of the Migration Process. In presenting this process the Migration Tools and Services that are available will be highlighted and placed within the perspective of the entire process. For more information on the tools refer to the paper MPE V to MPE XL Migration: Migration Tools in these proceedings.

## Migration Process Overview

The Migration Process consists of six stages that delineate the steps that are normally taken when migrating from an existing MPE V based HP 3000 system to a 900 Series HP 3000 system. These stages are defined at a high level since the specific steps in the migration of a given application is highly dependent upon the applications characteristics. Below is a brief description of the stages followed by more details in the next section.

These high level stages begin with *Education* on the Migration Stages, the Migration Tools and

	HP 3000	<b>R104/3</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Services that Hewlett-Packard is providing, as well as the high degree of compatibility that is provided by the 900 Series HP 3000 systems. This paper and past papers and presentations on various migration topics and the Migration Data Sheet are Hewlett-Packard's first steps in providing information on migration.

The second Migration Stage is *Predelivery Analysis and Planning*. In this stage the information learned in the *Education* stage is applied to a specific application. This is the first stage in which detailed analysis of Language, Database, and Network Migration and their relationship to an application is integrated. The end product of this stage is a migration plan.


The third stage is *Preparation*. In this stage preparation for the installation of the 900 Series system begins. This is also the first stage in which implementation of the migration plan starts. It primarily includes the steps that may be taken on an existing HP 3000 system in preparation for installation and migration to the 900 Series system.

The next stage is *System Installation*. This includes a series of steps that will be taken upon delivery and installation of the 900 Series system. In addition to the normal steps that are taken when a new HP 3000 system is installed, this step includes use of a new Migration Tool which duplicates the existing HP 3000 operational environment on a 900 Series system.

The fifth and sixth stages, *Compatibility Mode* and *Native Mode* execution, are both highly dependent upon the application and migration strategy. In *Compatibility Mode* execution, Object Code Compatibility plays an important part. Compatibility Mode primarily consists of running programs and using data from existing HP 3000 systems without modification of either. Source Code Compatibility is the key to *Native Mode* execution. It is in this stage that programs are recompiled using the new optimized Native Mode compilers. This stage may also include dual mode execution of programs.

### Education

The goal of the *Education* stage of the Migration Process is to learn about the Migration Process, Tools and Services as well as the new products being offered on the 900 Series systems so that this information can be applied when developing a migration plan for a specific application. Hewlett-Packard is doing a number of things that will provide the migration information that is needed. Information is available from papers and presentations which have been given at past

	HP 3000	<b>R104/4</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

INTEREX conferences in Detroit and Madrid. A Migration Data Sheet is also available and provides more information on migration to the 900 Series systems. (See the references section at the end of this paper for more information.) Field Sales Representatives have been provided with a Migration Sales Guide and we are now training our field support organization.

With the release of the Series 930 system a Migration Manual Set will be available. This set of self-paced training will include manuals for System Managers, Programmers, and General Users which will update them on (a) the differences between MPE V and MPE XL systems; and (b) information on significant new features. This manual set will also include a Migration Process Manual explaining the process and migration stages in detail.


A number of new MPE XL documents have already been released. The 900 Series HP 3000 General Information Manual has been released and includes an overview of new MPE XL based products as well as information on the Series 930 and the MPE XL operating system. An update to the HP 3000 System Configuration Guide has been released with supported configuration details for the Series 930.

The above information and other documents which will be released in the future should be reviewed as part of the *Education* stage of migration. These documents form a knowledge base by providing more information about MPE XL based products.

### **Predelivery Analysis and Planning**

The goal of the *Predelivery Analysis and Planning* stage is to produce a detailed plan for the migration of a specific application. This plan may include short term and long term goals for migration of the application. An example of a short term goal might include restoring a Segmented Library of SPL procedures in Compatibility Mode while the long term goal for these procedures might be to rewrite them in a language supported in Native Mode.

HP can provide consulting services which are tailored to a specific application or system migration. An HP consulting service that helps plan the migration of applications will be available. It will be delivered at the customer site by a trained HP Migration Specialist whose goal is to provide the methodology for optimal migration planning and to provide a migration plan for a specific application. This service will be delivered in two parts, Migration Orientation followed by Migration Analysis and Planning.

	HP 3000	<b>R104/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Other consulting services available on a time and materials basis can include assistance with the implementation of the Migration Plan. HP can also assist customers with Series II/III/30/33 systems in moving directly to a 900 Series system.

A Migration Toolset will be available. The Migration Toolset executes on existing MPE V based HP 3000s and helps analyze applications by identifying incompatibilities that may exist in your applications. The Toolset provides an automated way of determining the incompatibilities within an application, avoiding the need to manually search through source code. The PTAPE intrinsic, which reads a paper tape reader, is an example of an incompatibility that would be identified by the Toolset since it is not supported on MPE XL based systems.


The Toolset includes an Object Code Analyzer and a Run Time Monitor. The Object Code Analyzer will identify incompatibilities which may exist in programs and SL's. This tool will comprehensively scan individual or groups of programs and SL's and identify potential incompatibilities. The Run Time Monitor identifies incompatibilities which exist in applications as they execute. This tool will interrogate parameters passed to intrinsics for potential incompatibilities and log any detected incompatibilities for later reporting.

Based upon the information provided in the *Education and Predelivery Analysis and Planning* stages, a detailed migration plan can be created.

As with the installation of any new machine, preparation of the facilities must be taken into consideration as well as planning for which system each group of users will be using after the new machine is installed. Hewlett-Packard will be offering an extended return program that will allow customers to inexpensively operate an existing HP 3000 Series 6x/70 in parallel with the 900 Series system. The extended return program allows customers to keep the Series 6x/70 to be returned, beyond the normal 30 day return policy. The updated HP 3000 System Configuration Guide should also be consulted when planning for peripheral support on the Series 930.

### **Preparation**

The first implementation step in a migration plan is preparing for migration. During this stage everything that can be done on an existing MPE V/E based HP 3000 in preparation for delivery of

	HP 3000	R104/6
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

the 900 Series system should be performed. If any incompatibilities were found in the planning stage they should be isolated and if possible re-coded.

Language conversion activities that can be performed during this stage include migration of Fortran/V programs to Fortran 77/V as well as migration of Basic/V to HP Business Basic/V and COBOL 68/V to COBOL II/V. When possible SPL code should be rewritten in Pascal/V or C. If the SPL code cannot be rewritten on the MPE V system for performance or functionality reasons, it should be noted for future reference after the 900 Series system is installed. In most cases language migration could begin today on existing HP 3000 systems.

Other activities should include migration from Image/V to TurboImage/V, DS to NS Migration, and updating to the MPE V/E release that is recommended for migration.


### System Installation

Once the 900 Series system arrives, the *System Installation* stage begins. The primary goal of this stage is duplication of the existing HP 3000 operational environment on the 900 Series system. HP is providing new tools that provide increased functionality and ease of use in maintaining the operational environment as well as a migration tool which help migrate the existing HP 3000 environment to the 900 Series system.

A Directory Migration Tool will be available to assist in migration of the operating environment. This tool will migrate an MPE V accounting structure to an MPE XL based system. It will also migrate RIN Table Information and User Logging IDs as well as the UDC environment and Private Volume information.

A new System Generation utility, SYSGEN, replaces the MPE V SYSDUMP. The user interface for SYSGEN provides significant improvements over SYSDUMP. Specification of many devices with the same basic configuration and device address *only* changes, can be done with a single command for all of the devices instead of requiring a command for each device. Because most MPE XL tables are self expanding the system manager no longer needs to configure these table sizes.

The MPE XL Store/Restore command supports a 'TRANSPORT' option which allows MPE V

	HP 3000	R104/7
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

compatible store tapes to be created and read. This option facilitates the transfer of data between existing HP 3000 systems and 900 Series systems.

Since MPE V/E is not supported on the Series II/III/30/33 systems, these customers may also purchase consulting services to aid in migration of these systems to 900 Series systems during the *System Installation* stage.


### Compatibility Mode

Object code compatibility is provided via *Compatibility Mode*. 900 Series HP 3000 systems have a run time environment known as 'Compatibility Mode' which allows customer developed object code from the MPE V based HP 3000 family to run on 900 Series systems. Consequently, few changes, if any, are required to move these applications or data from existing HP 3000s to new 900 Series systems. While Compatibility Mode provides the ability to migrate applications to 900 Series systems quickly with little changes it also provides the ability to phase migration to Native Mode. This phased migration allows programs to execute in Compatibility Mode until they can be recompiled into Native Mode. By migrating to Native Mode, an application can take advantage of the best performance and new features of the HP Precision Architecture.

When a user runs a program on an MPE XL system the MPE XL Loader determines whether the program was generated using one of the new MPE XL compilers or with one of the MPE V compilers. If the program was generated with an MPE XL compiler it is called a Native Mode program since it uses the 900 Series instructions and addressing. If the program was generated using one of the MPE V compilers or was restored from an MPE V system it is considered a Compatibility Mode program.

If the MPE XL Loader determines that the program is a Native Mode program the program will be loaded and executed. If the program is a Compatibility Mode program, the MPE V HP 3000 Instruction Set Emulator will be invoked to emulate the program. The user does not need to know what kind of program is being run since this is determined by the MPE XL operating system.

Hewlett-Packard is also supplying an Object Code Translator to improve performance within

	HP 3000	R104/8
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

**Compatibility Mode.** The Object Code Translator will translate the MPE V HP 3000 instructions in a Compatibility Mode program or SL into 900 Series instructions and append the 900 Series instructions to the end of the program or SL file. A new command much like the existing compiler commands is used to perform the translation. The translator provides better performance when the program is run since the HP 3000 instructions do not need to be translated into 900 Series instructions at run time.


While providing increased performance for programs that are not recompiled into Native Mode, the Object Code Translator significantly increases the programs or SL's physical size. Besides providing increased performance by avoiding the decoding of instructions at run time, the Object Code Translator also optimizes the code. One of the side effects of optimizing and translating the code is that debugging becomes more difficult.

### Native Mode

The 900 Series HP 3000 systems have a primary environment known as 'Native Mode' which allows efficient accessing of the full power of the 900 Series new HP Precision Architecture. An existing application can be easily recompiled to Native Mode because new 900 Series compilers have been designed to provide source code compatibility with the rest of the HP 3000 family. These compilers will be released in phases and include HP FORTRAN 77/XL, COBOL II/XL, HP Pascal/XL, HP C/XL, HP Business BASIC/XL, HP Transact/XL, and HP RPG/XL.

Since a phased migration approach is possible, HP is supporting mixed mode applications. A mixed mode application is one which executes in both Compatibility Mode and Native Mode. This feature is provided via the MPE XL Switch Subsystem and allows Native Mode programs to call procedures which reside in Compatibility Mode SL's. It also allows Compatibility Mode programs to call procedures which reside in Native Mode executable libraries (the Native Mode equivalent of SL's). The Switch Subsystem consists of three new intrinsics which are used to perform these mode switches.

A common example of using the Switch Subsystem is a COBOL II/V application that calls SPL/V procedures located in an SL. The COBOL II/V application can be easily recompiled using the COBOL II/XL Native Mode compiler. The SPL/V procedures could be left in the Compatibility Mode SL and the recompiled COBOL application could continue to call these procedures via the MPE XL Switch Subsystem. (Note: There is a performance penalty incurred when switching modes over normal procedure calls.) In this manner the MPE XL Switch Subsystem plays a key part in providing a phased migration of applications.

	HP 3000	R104/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Hewlett-Packard will also be providing a Switch Assist Tool which makes use of the Switch Subsystem significantly easier. The Switch Assist Tool allows entry of information about the procedure that needs to be called and generates source code that makes the call to the proper Switch Subsystem intrinsic. This means that, in the above example, the COBOL II application will not need to be modified to make the calls to the Switch Subsystem.

Based upon user requests that HP support more international standards and as a goal in providing a single standard across HP computer systems, the HP Precision Architecture machines support the IEEE floating point standard. While converting HP 3000 floating point data to the IEEE format is optional, Native Mode programs will have better performance if floating point data is converted to the IEEE format since the Floating Point Coprocessor supports the IEEE standard. To convert from the MPE V HP 3000 floating point format to the IEEE floating point standard format, which is supported in Native Mode on 900 Series systems, a new Floating Point Conversion Intrinsic will be provided. This intrinsic allows 32, 64 and 128 bit floating point numbers to be converted between the HP 3000 floating point format and the IEEE floating point format.

### **Conclusion**

Migration from MPE V based HP 3000 systems to 900 Series HP 3000 systems may be performed in a phased manner with a high degree of object code and source code compatibility. Complete Migration Tools and Extensive Documentation are being provided in each of the stages to make migration to the 900 Series HP Precision Architecture machines as easy as possible.



### References

900 Series HP 3000 Computer Systems General Information Manual, Hewlett-Packard Inc.

HP Precision Architecture -- A New Perspective, Hewlett-Packard Inc.

Information Management HP 3000 Specification Guide, Hewlett-Packard Inc.

Migrating COBOL Programs to Spectrum: A Battle or a Breeze, by Steven J. Spence, INTEREX  
1986 Detroit Conference Proceedings

Migration Data Sheet, Hewlett-Packard Inc.

Migration Solutions for MPE XL by Lawrence J. Cagnoni and I. Janet Garcia, INTEREX 1986  
Detroit Conference

MPE V to MPE XL Migration: Migration Tools by R. Gregory Stephens, INTEREX 1987 Vienna  
Conference


Object Code Compatibility on Future HP 3000 Systems, by R. Gregory Stephens, INTEREX  
1986 Madrid Conference

Relational Technology -- A Productivity Solution, Hewlett-Packard Inc.

Using the MPE XL Link Editor, by Cary A. Coutant, INTEREX 1986 Detroit Conference  
Proceedings

### Biography

R. Gregory Stephens works for the HP Computer Systems Division as the MPE XL Migration Product Manager. He also worked in the MPE XL Operating System Support group within the Information Technology Group. He received his B.S. in Management Information Science from California State University at Sacramento and worked for Lawrence Livermore National Laboratory as a Computer Scientist on HP 3000 systems.

	HP 3000	<b>RI05/1</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Design of the HP 3000 Series 950  
 Peter Rosenbladt, Hewlett-Packard

### Specs

The HP3000 Series 950 is a ULSI-based superminicomputer operating at 6-8 MIPS. It uses a 128kB cache and includes a >1MFLOP Floating Point Coprocessor. Memory capacity is from 16 to 256 MB. Two 20MB/sec I/O channels are included which support four 6MB/sec Channel I/O Busses.

### Historical Background


The HP3000 Series 950 carries on the tradition of the HP3000 established in 1972 by providing a fully compatible software environment. On the inside, however, it represents a radical departure. The 16-bit microcoded stack architecture with code and data segmentation of the original HP 3000 gives way to a 32-bit RISC architecture with 48-bit address space, central register set, and paging. The Series 950 is a member of HP's new family of computers designed around the Precision Architecture introduced in early 1986. The technical realization of the 950 grew out of experiences gained during the development of the advanced NMOS process for the HP 9000/500 computer and the symmetric shared-memory multiprocessor structure employed by this machine. In the I/O area achievements from HP's Vision program have been incorporated into the system design.

### SPU Organization

The System Processing Unit (SPU) of the HP3000 Series 950 is organized around a high speed 64 bit pipelined System Memory Bus (SMB). Up to 2 Memory Control Units - each supporting 128MB of Memory in 16MB increments - and up to 4 symmetric processors can attach to SMB. In addition, up to 2 architecturally-transparent Bus Converters (BC) connect the SMB to two 32-bit synchronous I/O busses. Each I/O bus can support up to 6 I/O Channels, 2 of which are internal to the SPU bay, and 4 are usable with an I/O Extender Bay. Each I/O Channel can support 5 devices if on the SPU and 8 devices in the Extender. An Access Port into the machine allows remote console access for maintenance and diagnostic purposes.

### Processor organization

Each of the up to four identical processors in the SPU is organized around a Cache Bus that handles instruction and data fetches from cache as well as cache line loads and stores to memory. The replacement algorithm is random. The Translation Lookaside Buffer (TLB) is single-set split instruction/data. The Central Processing Unit (CPU) is implemented on a single ULSI circuit and features a 3-stage pipeline with instruction prefetch and overlapped loads, stores and branches. The Floating Point Coprocessor handles IEEE Standard Single and Double precision Adds, Multiplies and Divides as well as selected convert operations. The System Interface Unit contains the SMB interface logic as well as the cache coherency logic to allow multiprocessor operation.

	HP 3000	<b>R105/2</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Product Design

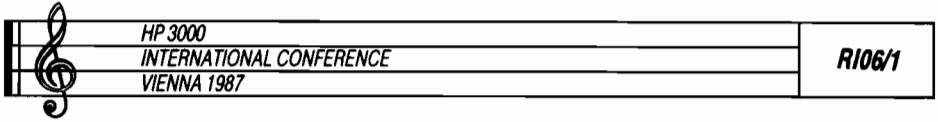
The HP3000 Series 950 has been designed to operate quietly, reliably, and efficiently in EDP room environments with or without raised floors. The SPU packaging consists of two 1 meter high bays, the Power System, and the Processor Bay. The Power Bay converts the AC input voltage to the required DC voltages. These are delivered to the Processor Bay via a copper laminated busbar. The Processor Bay contains the processor, memory, and IO systems. The key features of this design are the ULSI cooling system and the dense packaging of system components. The ULSI cooling consists of an innovative chip package design in which the chip is bonded directly to a copper-tungsten heat spreader which in turn is bonded to an aluminum heatsink. This arrangement allows for efficient cooling and very high overall packaging density. This, coupled with a very efficient machine organization, has resulted in a powerful product in a very small footprint. These features and many others incorporated into the product design should substantially bring ease of use benefits to the customer.

## Support Features

Coupled with its technology advances, HP provides diagnostic support to further lower costs of ownership to the customer. The HP3000 Series 950 has both local and remote support. This is accomplished by a three level strategy that uses a self test, standalone diagnostics and on-line diagnostics. Selftest verifies the boot path hardware required to load ISL (Initial System Load). The self test code is located in ROM. The System loads ISL, then the Operating System is booted. Once the Operating System is loaded, on-line diagnostics will report system failures. This can be run in either single or multiple user mode. If ISL is loaded and the Operating System does not boot, standalone diagnostics provide an additional SPU verification level on the processor, memory and I/O systems. Once these have been passed, the Operating System can be rebooted. All these diagnostics can be run locally or remotely. The remote user gains systems access, subject to locally controlled security restrictions, via the Access Port board. Through this approach HP provides secured remote diagnostics and a most effective means of support.

## Comparison with 1982 computer


The HP 3000 Series 950 is a reflection of the progress being made in computer design and engineering if one compares parts count, manufacturing cost, failure rate, and performance with a machine of the same class introduced 5 years ago.



**MPE V to MPE XL Migration: Migration Tools**

**by  
R. Gregory Stephens**

**Hewlett-Packard Company.  
19447 Pruneridge Ave.  
Cupertino, CA 95014**

	HP 3000	R106/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

### **Abstract**

This paper will provide an introduction to the various Migration Tools that support migration from MPE V based HP 3000 systems to 900 Series HP 3000. The tools addressed in the paper are all tools provided by Hewlett Packard. Each of the tools will be placed within the perspective of the migration stages.

### **Introduction**

Migration to HP 3000 Series 930 and Series 950 systems is based on full compatibility with the rest of the HP 3000 family. This compatible migration path protects your investment in HP 3000 hardware and software and provides you with a painless growth path to the high performance and capabilities of the 900 Series systems.


Hewlett-Packard has made a significant investment in providing a high degree of compatibility and ease of migration. This investment is exemplified in the areas of Object Code Compatibility, Source Code Compatibility, Database Compatibility, Network Compatibility, Operational Compatibility, as well as the Migration Tools and Services that are being offered.

This paper presents an introduction to the Migration Tools. In presenting the Migration Tools the role of each tool will be highlighted and placed within the perspective of the entire migration process.

### **Migration Process Overview**

If you have read the MPE V to MPE XL Migration Overview paper then the following description of the migration stages may be skipped.

The Migration Process consists of six stages that delineate the steps that are normally taken when migrating from an existing MPE V based HP 3000 system to a 900 Series HP 3000 system. These stages are defined at a high level since the specific steps in the migration of a given application is highly dependent upon the applications characteristics. Below is a brief description of the stages followed by more details in the next section.

	HP 3000	RI06/3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

These high level stages begin with **Education** on the Migration Stages themselves, the Migration Tools and Services that Hewlett-Packard is providing, as well as the high degree of compatibility that is provided by the 900 Series HP 3000 systems. This paper and past papers and presentations on various migration topics and the Migration Data Sheet are Hewlett-Packard's first steps in providing information on migration.

The second Migration Stage is **Preelivery Analysis and Planning**. In this stage the information learned in the **Education** stage is applied to a specific application. This is the first stage in which detailed analysis of Language, Database, and Network Migration and their relationship to an application is integrated. The end product of this stage is a migration plan.

The third stage is **Preparation**. In this stage preparation for the installation of the 900 Series system begins. This is also the first stage in which implementation of the migration plan starts. It primarily includes the steps that may be taken on an existing HP 3000 system in preparation for installation and migration to the 900 Series system.

The next stage is **System Installation**. This includes a series of steps that will be taken upon delivery and installation of the 900 Series system. In addition to the normal steps that are taken when a new HP 3000 system is installed this step includes use of a new Migration Tool which may be used to duplicate the existing HP 3000 operational environment on a 900 Series system.

The fifth and sixth stages, **Compatibility Mode** and **Native Mode** execution, are both highly dependent upon the application and migration strategy. In **Compatibility Mode** execution, Object Code Compatibility plays an important part. Compatibility Mode primarily consists of running programs and using data from existing HP 3000 systems without modification of either. Source Code Compatibility is the key to **Native Mode** execution. It is in this stage that programs are recompiled using the new optimized Native Mode compilers. This stage may also include cross-mode execution of programs.

## **Predelivery Analysis and Planning**

The goal of the *Predelivery Planning* stage is to produce a detailed plan for a phased migration of a specific application. This plan may include short term and long term goals for migration of an application. One of the key tasks in planning for migration is determining whether any incompatibilities exist in the migrating application. To help identify these incompatibilities a **Migration Toolset** will be available.

The Migration Toolset executes on existing MPE V based HP 3000s and helps identify incompatibilities that may exist in your applications. The Toolset includes an **Object Code Analyzer** and a **Run Time Monitor**. The Object Code Analyzer will identify incompatibilities which may exist in programs and SL's. This tool will comprehensively scan individual or groups of programs and SL's and identify incompatibilities which may exist. The Run Time Monitor identifies incompatibilities which exist in applications as they execute. This tool will interrogate parameters passed to intrinsics for potential incompatibilities and log any incompatibilities for later reporting.

Both tools identify two types of incompatibilities, those that apply only when an application is recompiled in Native Mode, and those that apply in both Native Mode and Compatibility Mode. Since the output of the tools is dependent upon an understanding of Compatibility Mode and Native Mode a description of these two modes of execution and their differences is in order.

### **Introduction to Compatibility Mode**

HP 3000 900 Series systems have a run time environment known as 'Compatibility Mode' which allows customer-developed object code from the MPE V based HP 3000 family to run on 900 Series systems. Consequently, no changes are required to move these applications or data from existing HP 3000s to new 900 Series systems.

When a user runs a program on an MPE XL system, the MPE XL Loader determines whether the program was generated using one of the new MPE XL compilers or with one of the MPE V compilers. If the program was generated with an MPE XL compiler, it is called a Native Mode program since it uses the 900 Series instructions and addressing. If the program was generated using one of the MPE V compilers or was restored from an MPE V system, it is considered a Compatibility Mode program.

If the MPE XL Loader determines that the program is a Native Mode program the program will be loaded and executed. If the program is a Compatibility Mode program the HP 3000 Instruction Set Emulator will be invoked to emulate the program. The user does not need to know what kind of program is being run since all of this is performed by MPE XL.

### Introduction to Native Mode

The HP 3000 900 Series systems have a second environment known as 'Native Mode' which allows efficient accessing of the full power of the 900 Series new HP Precision Architecture. An existing application can be easily recompiled to Native Mode because new 900 Series compilers have been designed to provide source code compatibility with the rest of the HP 3000 family. These compilers will be released in phases and include HP FORTRAN 77/XL, COBOL II/XL, HP Pascal/XL, HP C/XL, HP Business BASIC/XL, HP Transact/XL, and HP RPG/XL.

### Object Code Analyzer

The Object Code Analyzer (OCA) is a dialog-driven tool that allows the user to enter a list of programs and SL's to be analyzed for potential incompatibilities. Wild card specification of file names is supported, allowing a specification such as @.@.myacct. In which case the entire account 'myacct' will be searched for all programs and SL's which will then be analyzed by OCA.


The types of potential incompatibilities that can be identified by OCA include:

- References to MPE V Intrinsic which have changed
- References to uncallable MPE V procedures

The fact that OCA performs its analysis without running the program to be analyzed (versus RTM which performs its analysis on an executing program) provides several distinct advantages. The entire program file or SL is processed and the user can specify that the analysis take place when the load on the system is light and in batch mode if desired. The analysis is also not dependent upon MPE Files or Image Databases.

One of the disadvantages to this approach is that incompatibilities in parameters cannot be identified since the parameters passed to a given intrinsic cannot be identified until run time. It is for this reason that incompatibilities listed in the OCA report are referred to as 'potential incompatibilities'. References in the documentation will provide the information needed to determine the specific incompatibility related to a message in the OCA report.



	HP 3000	<i>R106/6</i>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

OCA will also generate a list of all of the files scanned in a given execution of OCA. Since OCA allows specification of an indirect file which contains a list of file names, a list generated by OCA or any other list of files can be processed by OCA by specifying the name of a file that contains a list. OCA also allows entry of additional external procedure names. This capability provides the ability to scan for calls to user written procedures that reside in an SL. This can assist in tracking down calls to user written procedures which may need to be modified.

After OCA has completed scanning, output in the form of a brief or detailed report will be generated. Both reports will list procedures that are called which have a potential incompatibility. It will also specify whether the incompatibility applies to Native Mode only or to both modes. A reference number will also be listed for each incompatibility. This number may be used to look up more information about the incompatibility in the documentation. This number is useful because it provides a standard incompatibility identification that is used in the documentation and between the tools that make up the Migration Toolset.

Both report formats also provide general information about the program. A list of capabilities that the program requires including whether the program or SL contains privileged segments is provided. This is valuable since privileged code will need to be reviewed carefully in light of the extensive differences between MPE V and MPE XL internals.

In addition to the information provided in the brief report the detailed report provides segment information (i.e. sizes, privileged segments, average segment size, entry points, patch area information), as well as the PB-relative locations of calls to resolved external procedures.

### Run Time Monitor

The Run Time Monitor (RTM) is a tool that detects incompatibilities in a program as it executes. When a program executes, any calls to intrinsics with potential incompatibilities are intercepted by RTM. RTM reviews the parameters being passed to the intrinsic and if a changed or unsupported parameter is being passed, the event will be logged to the system log file by RTM. A separate reporting program may be later run which will list the incompatibilities logged by RTM.

Besides the reporting program, RTM includes an SL that resides in the PUB.SYS group. When RTM is activated, the MPE V loader searches the RTM SL before searching the system SL for any external procedures. The RTM SL contains entries for intrinsics which have incompatibilities.

The types of incompatibilities that can be identified by RTM include:

- References to MPE V Intrinsic that are obsolete
- Intrinsic parameters that have changed or obsolete
- Programmatically executable commands that are obsolete
- Programmatically executable commands whose output has changed

The fact that RTM detects incompatibilities at run time has distinct advantages. Since the parameters are known, an intrinsic which has an incompatibility related to a specific parameter is not logged unless that specific parameter is passed to the intrinsic.

The disadvantage to this approach is that only portions of a program or SL that execute is analyzed. A few more words of stack space are also used in this approach since the parameters are effectively passed twice. Once to the RTM stub intrinsic and again to the actual intrinsic in the system SL.


RTM logs events to the system log file using system logging event 16, Program File Event. To log this event it must be enabled via SYSDUMP. RTM also includes a controlling utility which allows the control of the types of incompatibilities that should be logged. This capability allows you to individually view specific types of incompatibilities.

### Summary

The Object Code Analyzer and Run Time Monitor together provide complete and detailed analysis of programs and SL's for potential incompatibilities that should be considered during the Predelivery Planning stage of migration to the 900 Series HP 3000 systems.

### System Installation

Once the 900 Series system arrives the *System Installation* stage begins. The primary goal of this stage is duplication of the existing HP 3000 operational environment on the 900 Series system. To make duplication of this environment as easy as possible Hewlett Packard has developed the Directory Migration Tool (DIRMIG).

	HP 3000	<b>R106/8</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

DIRMIG will migrate an MPE V accounting structure to an MPE XL based system. It will also migrate RIN (Resource Identification Number) Table Information and User Logging IDs as well as the UDC environment and Private Volume information.

Before running DIRMIG, the MPE XL system should be up and running and configured using SYSGEN. An MPE V SYSDUMP tape must be created which will act as input to DIRMIG. To insure that migration using DIRMIG is as easy as possible the RIN and User Logging Identifier information should be up to date and all UDC files should be stored at the beginning of the store set.

MPE V private volumes will not be directly supported on MPE XL systems. DIRMIG will however generate the MPE XL VOLUTIL commands necessary to create the equivalent MPE XL Private Volumes. DIRMIG allows all or part of the MPE V directory structure to be duplicated on the MPE XL system. DIRMIG will only generate VOLUTIL commands for private volume directories which have been selected for migration.

### Compatibility Mode

As described earlier, object code compatibility with MPE V based HP 3000 systems is provided on 900 Series systems via *Compatibility Mode*. To provide increased performance without requiring recompilation HP is supplying an Object Code Translator.

While the MPE V HP 3000 Instruction Set Emulator that is used by default in compatibility mode is analogous to an interpreter (i.e. It decodes the MPE V HP 3000 instructions at run time), the Object Code Translator is analogous to a compiler. A new command, much like the existing compiler commands, is used to perform the translation. The Object Code Translator will translate the MPE V HP 3000 instructions in a Compatibility Mode program or SL into 900 Series instructions and append the 900 Series instructions to the end of the program or SL file. The translator provides better performance when the program is run since the MPE V HP 3000 instructions do not need to be translated into 900 Series instructions at run time.

Besides the increased performance that is achieved by avoiding the decoding of the MPE V HP 3000 Instructions at run time, the Object Code Translator (OCT) performs several other optimizations. The OCT provides significant performance optimization by eliminating unnecessary condition code, carry, and overflow tests. Since the OCT performs path code analysis during

translation it can determine whether or not tests are needed. The OCT can also store the top 8 words (16 bit) on the top of stack in 900 Series general registers. It can also calculate constants into the generation of 900 Series Instructions.

### Native Mode


Migration of applications to Native Mode will provide the full performance and features of the HP Precision Architecture. To take advantage of Native Mode programs must be recompiled using one of the new MPE XL Native Mode compilers. MPE XL compilers for high level HP 3000 languages will have a phased availability.

### MPE XL Switch Subsystem

In support of the phased migration strategy and because a Native Mode SPL compiler is not available, MPE XL will support mixed mode execution. This mixed mode capability is provided by the MPE XL Switch Subsystem and is exposed via three new MPE XL intrinsics. These new intrinsics provide two basic capabilities.

- **Native Mode to Compatibility Mode Switching.**  
This capability allows Native Mode programs to call procedures which reside in Compatibility Mode SL's.
- **Compatibility Mode to Native Mode Switching.**  
This capability allows Compatibility Mode programs to call procedures which reside in Native Mode XL's (Executable Libraries, the NM equivalent of SL's)

A common example of using the Switch Subsystem is a COBOL II/V application that calls SPL/V procedures located in an SL. The COBOL II/V application can be easily recompiled using the COBOL II/XL Native Mode compiler. The SPL/V procedures could be left in the Compatibility Mode SL and the recompiled COBOL application could continue to call these procedures via the MPE XL Switch Subsystem. In this manner the MPE XL Switch Subsystem plays a key role in providing a phased migration of applications.

	HP 3000	R106/10
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

In the example above the COBOL application that was recompiled can no longer call the target procedure directly. It must now call the appropriate MPE XL Switch Subsystem intrinsic which will invoke the target CM procedure on the COBOL program's behalf. It would be convenient if modification of the COBOL program, replacing calls to the target procedure with calls to the switch intrinsic, could be avoided.

To avoid modification of the COBOL program a Native Mode procedure could be written with a declaration that is identical to the target procedure. This NM procedure, called a **Switch Stub**, would then call the switch intrinsic to invoke the target CM SL procedure. Later, if the CM SL procedure is rewritten as part of the next stage in migration, the Native Mode Switch Stub can be replaced by a version of the CM SPL target procedure that has been recoded in Native Mode. In writing the Switch Stub we have avoided any modification of the COBOL program throughout all of the stages of the migration.


### Switch Assist Tool

Hewlett-Packard will be providing a Switch Assist Tool which makes use of the Switch Subsystem significantly easier. The Switch Assist Tool allows entry of information about the target Compatibility Mode procedure that is to be called and generates source code which makes the call to the proper Switch Subsystem intrinsic so that the Switch Stub does not need to be written from scratch.

The Switch Assist Tool incorporates a VPLUS interface that prompts for information about the target procedure such as the name of the procedure, the number of parameters, the type and length of each parameter, etc. After all of the information has been entered a status screen is displayed which reports on the status of the source code generation. The Switch Assist Tool generates Pascal source code.

### Conclusion

Migration from MPE V based HP 3000 systems to 900 Series HP 3000 systems may be performed in a phased manner with a high degree of object code and source code compatibility. The Migration Tools addressed in this paper are used from the Planning stage of migration to Native Mode execution and greatly simplify migration to MPE XL based HP 3000 systems.

	HP 3000	<i>R106/11</i>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## References

900 Series HP 3000 Computer Systems General Information Manual, Hewlett-Packard Inc.

HP Precision Architecture -- A New Perspective, Hewlett-Packard Inc.

Information Management HP 3000 Specification Guide, Hewlett-Packard Inc.

Migrating COBOL Programs to Spectrum: A Battle or a Breeze, by Steven J. Spence, INTEREX  
1986 Detroit Conference Proceedings

Migration Data Sheet, Hewlett-Packard Inc.

Migration Planning Assistance Data Sheet, P/N 5954-8629 (1/87) Hewlett-Packard Inc.

Migration Solutions for MPE XL by Lawrence J. Cargnoni and I. Janet Garcia, INTEREX 1986  
Detroit Conference

MPE V to MPE XL Migration: Migration Tools by R. Gregory Stephens, INTEREX 1987 Vienna  
Conference

Object Code Compatibility on Future HP 3000 Systems, by R. Gregory Stephens, INTEREX  
1986 Madrid Conference

Relational Technology -- A Productivity Solution, Hewlett-Packard Inc.

Using the MPE XL Link Editor, by Cary A. Coutant, INTEREX 1986 Detroit Conference  
Proceedings

## Biography

R. Gregory Stephens works for the HP Computer Systems Division as the MPE XL Migration Product Manager. He also worked in the MPE XL Operating System Support group within the Information Technology Group. He received his B.S. in Management Information Science from California State University at Sacramento and worked for Lawrence Livermore National Laboratory as a Computer Scientist on HP 3000 systems.

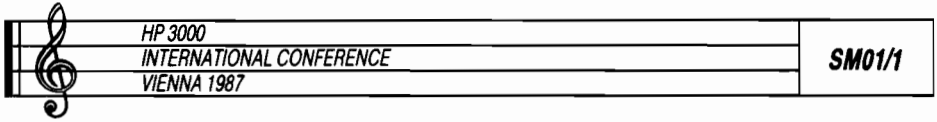


*HP 3000*

*INTERNATIONAL CONFERENCE*

*VIENNA 1987*





## **Converting to IBM – Are You Sure?**

**Donal Barksdale**

**Spectra-Physics  
3333 North First Street  
San Jose, CA 95134-1995  
(408) 946-6080**





## INTRODUCTION

### OVERVIEW

This paper is intended to focus on how to evaluate when or if you should change hardware. The assumption of the paper is that the decision to change computer hardware vendors is a business decision, and not a technical one. The process of reviewing the hardware decision is discussed using the business requirements as the catalyst. From here, the current hardware is reviewed AFTER analyzing the requirements. After reviewing the current vendor against 1, 2, and 5 year requirements, other vendors are measured using the same criteria. In addition, hidden costs associated with changes in hardware are identified.

Although this paper is directed toward HP users contemplating changes to IBM hardware, the evaluation process can be universally applied for any hardware evaluation.

The case study examines an actual company's decision to change hardware, and the effects of that change.

### SYMPTOMS

There are many symptoms that signal the mood and climate for a hardware change. Some of them are rational, others are not. Among a few of the more common ones:

- The new company president used IBM hardware at his last company and has decided to use the same system here.
- The manufacturing VP prefers this special software that only runs on IBM computers.
- Our current hardware vendor does not offer any larger machines, so we will need to change to IBM computers.
- We need to change hardware so that our systems can be compatible with headquarters.

These statements, of course may or may not be present in your situation. They may be good or bad reasons for change. If they are present, however, they will affect the entire process and it may not be possible to change management's perception overnight. The process of examining the business environment will yield satisfactory results to all and also provide valuable insight to what you SHOULD be working on!

The process of deciding on a vendor consists of the following steps:

- CURRENT MIS ASSESSMENT
- DETERMINE CURRENT HEALTH OF THE COMPANY
- REVIEW THE COMPANY FIVE-YEAR PLAN
- CURRENT HARDWARE
  - VENDOR PRODUCT LINE REVIEW
  - FIRST YEAR REVIEW
  - SECOND YEAR REVIEW
  - THIRD THROUGH FIFTH YEAR REVIEW
- PROPOSED HARDWARE #1 ASSESSMENT
  - VENDOR PRODUCT LINE REVIEW



THIRD THROUGH FIFTH YEAR REVIEW  
PROPOSED HARDWARE #2 ASSESSMENT  
VENDOR PRODUCT LINE REVIEW  
FIRST YEAR REVIEW  
SECOND YEAR REVIEW  
THIRD THROUGH FIFTH YEAR REVIEW  
COMPARISONS  
RECOMMENDATION

Each of these areas are discussed in the detail that follows.

PROCESS

BUSINESS CLIMATE

CURRENT MIS ASSESSMENT


This should only be an assessment and not a recommendation. You are probably very aware of the your total environment. Generally, it is possible to determine a lot about the company from the climate in the MIS department. Review manpower loading schedules, short term projections, and staffing issues. Of course, a very important part of the review is an assessment of performance and capacity issues. Be aware of the true state of the environment. Are you running old software? How are the problems affecting the business? Have you had a formal capacity and performance review? Are you aware of problems that can be fixed? For example, did you know that it is possible to expand your system's memory on a Series 70 to 32Mb? If you are running multiple CPUs, can you shift the workload? Are you impacting the business because of the hardware, or do you have a staffing problem?

DETERMINE CURRENT HEALTH OF THE COMPANY

This part includes an evaluation of financial factors. Since most MIS departments are part of the financial community, they tend to be well informed about the health of the company. Is the industry showing a downturn? Is the company a target for a takeover bid? Are there major staff reductions planned? Is there stability in the upper management team? The answer to most of these questions will have a significant impact upon the approach that you will need to take when reviewing the hardware situation.

REVIEW THE COMPANY'S FIVE YEAR PLAN

You must know the short and long term goals for the company. Review the data for the entire corporation and all divisions. What are the growth rates for each division? How will these business needs impact the needs for MIS services? Is there a focus on centralization, distributed processing, departmental processing, etc.? How are you providing MIS services to each group? What does the five year plan suggest about future allocation of services compared with today's allocation? Are business needs defined that clearly indicate capabilities not

	HP 3000	<b>SM01/4</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

currently supported (artificial intelligence, decision support systems, etc.)? Has the five-year plan been adopted? How is the company currently progressing when measured against the previous five year plan? Try to prioritize needs with upper management into 1, 2 and 5 year categories.

You may wonder how to proceed if there is no five-year plan. If this is the case, then you would be best advised to proceed very carefully before considering changing hardware. Of course, there may still be companies whose growth rates would suggest that the risks of changing hardware are small. It would be a good idea for you to review these cases with upper management.

#### CURRENT HARDWARE ASSESSMENT

##### PRODUCT LINE REVIEW

Of course, it only makes good business sense for you to spend time looking at your current vendor's product offering carefully. After all, you have a tremendous investment of time, hardware, training, and software.

By now, you should be familiar with your company's total environment, since you know capacity and performance data, the financial situation, and have been made aware of those items that make up the five-year plan. It is necessary to now review the total offerings from your current vendor. The review should consist of:

- What are the high-end capabilities?
- What are the low-end capabilities?
- Where does your company currently stand with installed hardware?
- If you are at the high-end, does the vendor offer you the option to cluster machines, and share devices such as discs and printers?
- Can the software be migrated across the entire product line?
- Connectivity to other mainframes?
- What future technology can you expect?
- What kind of service and support will you receive in the future?
- If there are new offerings, can you upgrade to the new products and preserve your current software?
- What is the financial health of your current vendor?
- What other technology advances outside of the mainframe are possible?

If we examine the answer to these questions in relation to HP, the answers will seem to be as follows:

**What are the high end capabilities?** The current available high end Series 70 is the only offering today. By June, it appears that a Series 950 will be offered. The capabilities are yet to be determined. The processing power compared to the existing offerings is expected to be 2 times greater.



What are the low-end capabilities? The current offerings include a micro 3000 that permits you to use existing software without modification.

Where does your company currently stand with installed hardware? You should inventory the total installed base within your company.

Does the vendor offer you the option to cluster machines, and share devices such as discs and printers? The current HP offerings do allow clusters. There is currently no direct sharing of peripherals between CPUs.

Can the software be migrated across the entire product line? The answer appears to be yes for the current and future announced products.

Connectivity to other mainframes? Yes, there ways to connect to IBM through communication controllers and DEC equipment through X.25 interfaces.

What future technology can you expect? Risc based architectures will become more widespread among the product lines.

What kind of service and support will you receive in the future? Probably no less than the current levels, but probably less than the volume that IBM has offered in the past and will probably continue.


What is the financial health of Hewlett Packard? Most users and industry observers do not currently have any concerns about HP's long term financial health.

What other technology advances outside of the mainframe are possible? The advancement of the PC is certainly an area that should not be overlooked. It may be possible to totally eliminate the need to change hardware, based upon redesigning systems and taking advantage of the increasing number of applications that allow you to co-exist with mainframes. HP, as well as other vendors such as Gateway System's product called Synergist are now looking at ways to reduce the load on mainframes.

These are just a few of the basic questions. The most important ones will relate to the actual business needs that have been identified previously. It is still possible that your needs cannot be met even though the obvious questions seem to be answered.

#### FIRST YEAR REVIEW

The one year needs should be very obvious by now. It should not be difficult to determine where you stand for the short term.

	HP 3000	<b>SM01/6</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

You will need to review the needs over the next year if you are planning a major upgrade with the existing vendor.

A typical list should include at least the following key areas:

- Business Systems
- Hardware
- Software
- Staff

#### Business Systems

Identify all requirements that would be needed during the next year. Remember, if you are planning a major upgrade, use two year requirements to size the initial upgrade configuration. This will give you a good idea of how your investment will be after you meet the one year needs. It will also let you know if you will be making the same decision next year.

Establish hardware and software requirements dictated by the applications selected. It is very useful to separate this data from the normal business systems.

Set priorities for implementation.

Determine BUSINESS SYSTEMS Costs, including:

- All purchased software or depreciation expense.
- Staff labor expenses
- Outside Contractor Costs
- Training
- Overlapping software cost
- Staff turnover

#### Hardware:

Identify initial hardware configuration based upon business systems selected.

Identify site considerations/changes required to accommodate the new hardware.

Review operational procedures required to use new hardware including:

- hardware maintenance schedules
- operator training

Determine HARDWARE costs, including:


- Hardware or depreciation expense of CPUs, controllers, modems, printers, terminals, etc.
- Annual Hardware maintenance costs
- Site rearrangement costs such as expansion, special line condition equipment, etc.
- Overlapping hardware costs during conversion.

#### Software:

Identify all operating system components.

Determine SOFTWARE costs, including:

- Annual operating system, utility, and development support costs.
- One-time software installation costs for operating system, utility, and development software.

	HP 3000	<b>SM01/7</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

**Staff:**

- Identify changes needed in staffing levels.
- Identify training needs for current staff.
- Determine STAFF costs, including:
  - Annual costs during and after conversion of planned staffing levels
  - Recruiting costs
  - Training, including travel

**Risk Assessment**

This should include your evaluation of the ability to meet the established goals in the Business Systems area. It would also include any areas that you have identified, whether they would be performance, service, or capacity issues. Talk to other companies who have made similar conversions.

**Summary of costs**

The total estimate of costs should be evaluated for the year.

**SECOND YEAR REVIEW**

The second year look must be a little more critical. By looking at your requirements alone, you will surely miss some things. You must be able to try to reduce any high risk factors that may cause you to make an inappropriate decision. Here are a few other things that you may want to consider:

- You should attempt to assess how the business will do if it meets its 2 year goals. Just as important, you should weigh the options if the goals are not met. If your current vendor (HP) will probably support you for one year and the second year now seems uncertain, you may want to consider some other factors. What is the level of confidence from upper management? Is the industry uncertain? If the second year of the company is uncertain, you may want to wait. On the other hand, if the growth has been 20 to 30 percent each year, you will certainly want to make sure that HP can support you based upon whatever philosophy your company will be following (centralized, departmental, etc). Consider the fact that you will PROBABLY NEED 2 YEARS TO CONVERT HARDWARE AND SOFTWARE SYSTEMS, with the first year being the most difficult. It will be very difficult to cut back a conversion midstream if the business turns sour. You will need to figure in overlapping costs in the first year for hardware, as well as training, recruiting, etc. Establish costs the same as in the first year.

**THIRD THROUGH FIFTH YEARS**

The third through fifth year projection should be used as just a way to project a growth curve, assuming the optimal conditions. It may be one way to project the ability of a vendor



to grow with your company. You should expect that there is a reasonable chance that you will still be with the same vendor at that point. Unless the vendor can support your company with a growth rate of at least 25% each year, it would not be wise to change vendors. For the years 3,4,5 you should compute two different costs:

First, compute the costs of all of the 4 categories associated with maintaining a no growth environment from the second through fifth years. This will give you some indication of what things would look like if the business did not grow. Compare it with the other vendor estimates.

Next, compute the costs for all of the 4 categories, using the growth from the five year plan.

#### VENDOR #1 HARDWARE ASSESSMENT

##### PRODUCT LINE REVIEW


This is an important part of the process of reviewing other vendors. You should review the breath of hardware and product offerings. This is merely an introduction to the alternative vendor. The actual mapping of the vendor to business requirements comes later, so try not to provide solutions before the needs have been taken into account. A minimum review should include:

- What are the high-end capabilities?
- What are the low-end capabilities?
- Can the software be migrated across the entire product line?
- Connectivity to other mainframes?
- What future technology can you expect?
- What kind of service and support will you receive?
- Are there upgrade paths when expanding?
- What is the financial health of the vendor?

It is very misleading to spend a lot of time looking at MIPS. Your main interest will be in throughput using the software you have selected and the ability to expand when you need it. It is probably more useful to understand relative performance among machines within a single vendor's offering. Get some feel for the costs, although you will not be able to determine real numbers until you review the needs for the first year. In those instances where the alternate vendor offers products by the same vendor, you will have the advantage of a better comparison. This is becoming more common as third party vendors expand across multiple hardware environment. Some examples of this are Cognos, and Ask Computers running similar software on HP and DEC.

Let's take a look at IBM and DEC.

Product line offerings

	HP 3000	<b>SM01/9</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

**IBM**

Few would argue that IBM has a long line of processors capable of handling large data processing needs. They include:

30xx Series High-end  
 43xx Series Mid-range  
 9370 Series Low-end  
 System 36,38 Low-end

**DEC**

With some of the recent announcements, DEC has expanded the depth of its product line. It offers:

897x,8800	Series High-end
8550,8600,8650,8700	Mid-Range
8200,8300,8500	Series Low-end
Vaxmate, MicroVaxII	Series Low-end

**Connectivity to other mainframes?**

**IBM**

SNA is the method that IBM has chosen to define the connectivity to its systems and others have created products to permit access to their mainframes. IBM has recently begun interfacing X.25 capabilities with SNA to broaden its appeal.

**DEC**

Dec has long offered connectivity to IBM and other machines via products like Decnet and its X.25 product offerings.

**What future technology can you expect from IBM?**

With the introduction of the 9370, IBM has increased its product offerings for those customers not needing large processor power. This trend will likely continue as they attempt to move to the minicomputer market.

**DEC**

The introduction of its line of 8974 and 8978 machines indicates a trend to providing unlimited growth for its customer base. The increased focus on commercial applications will continue.

**What kind of service and support will you receive in the future from IBM?**

The level of service and support is expected to continue, as they continue to discourage third party hardware maintenance.

**DEC**

The service level is generally rated as acceptable.

**What is the financial health of the company?**



IBM

Profits have certainly been higher, but IBM is a healthy company.

DEC Profits were up substantially in 1986 and are expected to continue in 1987.

It is also a good idea to contrast the actual working environments of the new vendor (IBM, DEC, etc) against your existing vendor (HP). It is worth reviewing the environments in further detail. Review the staff skills needed, operating system interface, compatibility in software across the product line, number of people needed for support, availability of software, etc.

There are many other distinctions between these two vendors, and you should spend some time focusing on those areas that will affect your business. Certainly, each system can perform the same functions as the other with differences, depending on processing requirements.

#### FIRST YEAR REVIEW


This is the biggest phase in determining the benefits of changing vendors. It is crucial, because it is here that the company will experience the largest variance in costs, and changes to the entire company. The business requirements would be the same as if there were no hardware changes, so try to be objective. You will need to review the needs over the 1 year period, as well as figure in the needs that are already met.

The same steps would apply as in looking at the first year for your existing hardware vendor. You should gather all data for:

- Business Systems
- Hardware
- Software
- Staff
- Risk Assessment
- Summary of costs

#### SECOND YEAR REVIEW

In some instances, it may be useful to combine business requirements for two years since the software must be changed. This will vary depending on the complexity of the needs that were

	HP 3000	<b>SM01/11</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

initially defined. It should be expected, however, that there will continue to be needs for the 2 second year as a result of implementation issues related to changing software.

In either case, the same cost data should be identified for the key areas of Business Systems, Hardware, Software, and Staff. If you have made a good choice, then the hardware selected should be capable of supporting your expected requirements with a minimum of additions.

### THIRD THROUGH FIFTH YEARS

The fifth year plan with a new vendor is similar to the situation of changing vendors. It should be assumed that this would also just be a model for the optimum situation. By using different growth rates, it is possible to estimate your new environment if things change. For the years 3,4,5 you should compute two different costs:

First, compute the costs of all of the 4 categories associated with maintaining a no growth environment from the second through fifth years. This will give you some indication of what things would look like if the business did not grow. Compare it with the original vendor estimate and others that you may be evaluating.

Next, compute the costs for all of the 4 categories, using the growth from the five year plan.

### MULTIPLE HARDWARE ASSESSMENTS

Multiple hardware assessments should each proceed the same as the previous comparison with IBM and DEC.

### COMPARISONS

This will typically take the form of a cost chart with brief summaries of the business systems with each vendor reviewed according to their ability to meet the requirements. Each year should be contrasted and the total expected costs compared over a five year period.

Be aware that cost will not always be the deciding factor. If a company is growing at a steady rate, then it is expected that the growth will sustain the added costs. Each company will want to establish acceptable costs based upon sales volumes.

### RECOMMENDATION

The actual recommendation should occur after a thorough review of the facts, along with some process of reducing high risk factors. There will be a need to gain a substantial commitment from the entire company if it is decided to change vendors.



## SUMMARY

In reviewing the process of changing hardware, the basic checklist must at least be:


- Assess the current MIS environment first. Do not recommend solutions at this time.
  - Understand the existing financial situation of the company.
  - Understand the priorities established by the five year plan.
  - Review the current vendors full product offerings.
  - Compare 1 year business requirements with existing hardware or next 2 year business requirements with any upgrade with the existing vendor.
  - Compare second year business requirements with existing vendor.
  - Estimate needs for years 3,4,5 assuming planned growth according to the five year plan.
  - Estimate needs for years 2,3,4,5 with no growth.
- For each Vendor under consideration:
- Review the proposed vendor's full product offerings.
  - Compare 2 year business requirements when sizing initial configuration.
  - Compare second year growth requirements with the new vendor.
  - Estimate needs for years 3,4,5 assuming growth according to the five year plan.
  - Estimate needs for years 2,3,4,5 with no growth.
- Make recommendation

It should be noted that to change vendors or not should be based upon the ability of the vendor to support the growth of the business and not vendor loyalty, or purely cost.

## CASE STUDY

### OVERVIEW

This is the profile of a large successful electronics company that is part of a multinational corporation. Sales for this subsidiary were in excess of \$130 million at the beginning of the project. The company had multiple plant locations, including foreign countries. This study will look at 2 years of actual implementation data.

	HP 3000	<b>SM01/13</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## SYMPTOMS

These were the symptoms that existed.

-There were many complaints that the existing software was inadequate for the business in virtually all areas.

-An almost total change in the upper management of the corporation, resulted in increased requirements being placed upon the existing hardware, software and staff.

-Expectations were high that there would be growth in all market areas.

## PROCESS

### BUSINESS CLIMATE

### MIS ASSESSMENT

The then assessment of the MIS organization was favorable from the user community in the area of support. Since the systems were very old, user requests consisted of major modifications and enhancements. The work load, as you would expect was very heavy, and many contractors were used to help meet deadlines. The hardware environment consisted of:

2 HP Series III

2 HP Series 68

This total hardware mix provided all of the commercial data processing support for the corporation. There was a staff of about 40 at that time.

The performance and capacity of the systems seemed to indicate that there were few alternatives left except to replace or upgrade the hardware.

### HEALTH OF THE COMPANY

The general health of this division was not profitable at the time, but since this was a small division with great promise, these early years were not perceived to be of particular concern. As stated before, many of the financial problems were attributed to the systems and the parent company appeared to believe that things would change soon. With the new management team, there was a focus toward centralization of functions.

### THE COMPANY'S FIVE YEAR PLAN

As expected, there was indeed a very ambitious plan in place to make the company an \$800 plus million company in 5 years. The major systems requirements were identified and consequently the decision was made to change hardware vendors. The major reasons for the changes were to be compatible with the parent company (IBM compatible), and because at the time, there was no high-end machine from HP, and the software needed replacing.

CURRENT HARDWARE ASSESSMENT

PRODUCT LINE REVIEW

The Hardware offerings from HP were reviewed, including a visit with the new development group. The answers to the questions at that time(early 1984) are shown.

- What were the high-end capabilities? The Series 68 was the only foreseeable offering. HP was recommending adding an additional processor.
- What were the low-end capabilities? The Series 30 was then one of the low end offerings.
- Where did the company stand with installed hardware? At that time, the company was using a multiple CPU approach, with additional hardware being the only possible expansion path.
- Did the vendor offer you the option to cluster machines, and share devices such as discs and printers? The company was using DS for host connectivity, but it did not prove satisfactory over long distances.
- Could the software be migrated across the entire product line? Yes.
- Connectivity to other mainframes? Yes, to HP and IBM machines.
- What future technology could we expect at that time? There were no new product offerings expected within the next year.
- What kind of service and support did we expect to receive in the future? There were no concerns in this area.
- Could we upgrade to the new products and preserve our current software? Yes.
- What was the financial health of HP? Stable.

1 YEAR REVIEW

Business Systems


All requirements were identified. The initial business systems requirements for two years were used to size the initial new hardware configuration.

Hardware and software requirements dictated by the applications were estimated. There were no final vendor selections made before selecting the hardware. Priorities were established for implementation.

The applications selected were:	One time cost
General Ledger	\$130,000
Accounts Payable	\$110,000
Accounts Receivable	\$120,000
Human Resources	\$110,000
Payroll	\$120,000

Costs were estimated including:

Depreciation costs of software(5 years)	\$119,000/year
First year prepaid maintenance	\$64,000
<b>Business Systems First year estimate</b>	<b>\$183,000</b>

	HP 3000	<b>SM01/15</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

- No overlapping costs for HP software were determined due to the fact that all software has been developed in house.

**Hardware:**

Initial hardware configuration based upon expected business systems was determined.

Site considerations/changes required to accommodate the new hardware were considered.

Operational procedures were developed to use new hardware including:

- hardware maintenance schedules
- operator training

The HARDWARE costs were:

- Hardware expenses of new IBM equipment \$300,000/year
- Hardware maintenance of IBM equipment \$38,000/year
- Computer room expansion \$100,000
- HP hardware expense \$120,000/year
- HP maintenance \$43,000/year

HARDWARE First year \$601,000

**Software:**

Identify all operating system components.

The SOFTWARE costs, including:

- Annual operating system, utility, and development support costs

- IDMS software purchase(3 years) \$54,000/year
- Roscoe/Librarian purchase(3 years) \$40,000/year
- Syncsort purchase \$2,500/year
- Tape Mngmt/Job Restart (3 years) \$18,720/year
- IBM other maintenance \$123,000/year
- IBM one-time charges \$20,000
- HP other maintenance \$17,000/year

SOFTWARE FIRST YEAR \$275,220

**Miscellaneous**

Not included are supplies, paper, storage space for manuals, compensation, data communications, contract labor, and training.

**Summary of costs**

The total estimate of costs should be evaluated for the year

BUSINESS SYSTEMS	First year	\$183,000
HARDWARE	First year	\$601,000
SOFTWARE	First year	\$275,220
Total		\$1,059,220

These numbers do not represent the entire expense budget, but do outline the major items involved in a conversion.

## 2 YEAR REVIEW

### Business Systems

The applications selected the first year were:

- General Ledger
- Accounts Payable
- Accounts Receivable
- Human Resources
- Payroll

In addition, the second year included:

Manufacturing Systems \$300,000

Costs were estimated including:

Depreciation costs of software(5 years)	\$119,000/year
Depreciation cost of Manufacturing S/W	\$60,000/year
Second year maintenance	\$87,000

Business Systems second year \$266,000

- No overlapping costs for HP software were determined due to the fact that all software had been developed in house.

### Hardware:

Costs, were determined including:

- Hardware expenses of new IBM equipment \$300,000/year
- Hardware maintenance of IBM equipment \$38,000/year
- HP hardware expense \$120,000/year
- HP maintenance \$43,000/year

HARDWARE second year \$501,000

### Software:

- IDMS software purchase(3 years) \$54,000/year
- Roscoe/Librarian purchase(3 years) \$40,000/year
- Syncsort purchase \$2,500/year
- Tape Mngmt/Job Restart (3 years) \$18,720/year
- IBM other maintenance \$123,000/year
- HP other maintenance 17,000/year


SOFTWARE second year \$255,220

### Miscellaneous

Not included are supplies,paper, storage space for manuals,salaries,data communications costs,etc.

### Summary of costs

BUSINESS SYSTEMS	First year	\$266,000
HARDWARE	First year	\$501,000
SOFTWARE	First year	\$255,220
Total		\$1,022,220

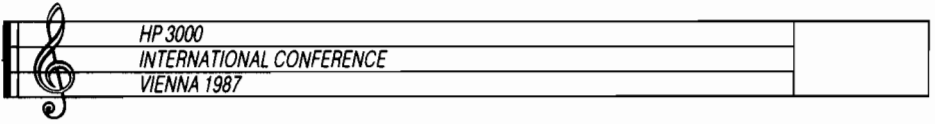
	HP 3000	SM01/17
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

These numbers do not represent the entire expense budget, but do outline the major items involved in a conversion.


#### SUMMARY

At this point, the company has experienced continued reductions in its market share. After carefully considering whether to reverse the conversion, the company has continued to proceed with the conversion effort.





HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

	HP 3000	SM02/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Predicting System Performance  
by Larry Kemp, HP Bellevue, WA, USA

System performance analysis has traditionally been a highly technical subject. Performance analysts given elaborate analyses of I/O counts, CPU states, and other operating system execution statistics to explain performance. While these internal measurements give indications of where to tune an application, they do not directly address the resulting performance criteria of response times and system throughput.

This paper presents techniques of answering some fundamental performance questions: What is a reasonable response time for an application? What will happen to response time and throughput when additional terminals and/or applications are added? What percentage of the usable system capacity is currently being consumed? For distributed systems, how will the application performance be affected by different hardware configurations? These are system management questions that focus on results.

I will present two techniques of predicting application performance: benchmarking, where an application is physically simulated; and modelling, where an application is mathematically simulated. These are techniques for analyzing *existing* applications. The "gut feel" approach will still be around for predicting applications that have not been written yet.

**Measurement Definitions.**

There are three fundamental measurements required for benchmarking and modelling: response time, transaction volume, and batch throughput. These variables can be considered the "dependent" variables; that is, when some portion of the environment is changed (such as workload or system configuration), then the effects can be measured using these criteria.

Users generally have good ideas of measurement criteria, but unfortunately those definitions do not always translate into easily quantifiable measurements. Both benchmarking and modelling require measurements: in benchmarking the results must be measured, while in modelling the current system must be measured.

Business systems analysts tend not to think in computer terms. They tend to think in business units. For instance, sales orders per day, or numbers of customer inquiry terminals. And transactions follow the same type of definitions. One performance analyst task is to translate *business* transactions into *computer* transactions. Normally there are one or more computer transactions for each business transaction. And furthermore, some business transactions are more or less complex than others.

For instance, if sales orders are increased by twenty percent, or fifteen customer inquiry terminals are added, how much will the total number of terminal transactions rise?

The raw data that a performance analyst works with is in computer transactions. An HP3000 transaction occurs any time the enter key or function key is pressed while in block mode, or any time the return key is pressed while in character mode. Hence the following technical definition:

*A Transaction occurs any time that a terminal read is satisfied.*

In other words, a transaction occurs any time that a user is required to wait for a response from the system.

The amount of work required per transaction can vary tremendously with application design. The amount of work required to process a character mode entry of a single field is likely to be significantly different than the work required to process a full screen entry in block mode, since a block mode screen consists of multiple fields. Consequently, comparisons of character mode and block mode applications will need to take this into account.

Response time is another area that is subject to interpretation. Ultimately, response time is the time from when a transaction is initiated to when the system is ready to accept the next transaction. This has the sometimes undesirable effect of including data communications time along with computer system response time.


In the interest of taking data communications out of the measurement, one solution is to measure the time from initiation to until the first character is output from the system. Unfortunately, this technique is defeated by applications that write out a message before processing the transaction. So here is the compromise definition:

*Response time is the time between terminal read completion and the last terminal write initiation before the next terminal read.*

This technique is in fact a compromise, in that it does take in to account data communications overhead for applications that write out multiple lines of data for each transaction. It is also prone to bias from user DC1/DC3 activity (from control-S and control-Q).

Response time is averaged against all actual terminal activity. Simple colon-prompt activity is only included to the extent that it happens on an actual system. If desired, response time could be measured against specific applications or transaction types.

Batch throughput is normally measured in terms of elapsed time to process a given workload. Since batch is a continual process (as are interactive terminals) and batch programs have widely varying

	HP 3000	SM02/3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

completion times, I prefer to specify batch throughput in normalized units. For example, if batch under one configuration may execute in 1 unit of time, while batch under another configuration may execute in .7 units of time.

Now, for the independent variables. A good experiment involves changing one parameter at a time, and recording the results. The same type of analysis should apply to predicting system performance. The variables to be changed include: number of terminals, think time between transactions, CPU type and speed, quantity and speed of disc drives, and software bottlenecks.

### Benchmarking.

Benchmarking involves live simulation of an application. This is a simplistic task for batch. For interactive applications, this involves either live operators, or computer simulation. In either case, a "script"; which is a set of input data for each terminal.

Benchmarking is a brute force operation. It is expensive in that it requires creating the application and scripts. And it requires access to hardware configurations and people. But *benchmarking does give results that can be compared between different hardware systems.* And it does not rely upon mathematical or procedural assumptions.

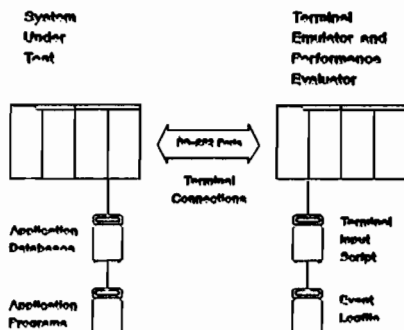
On the negative side, benchmarks are crude measures of the actual environment. The necessity of manually creating terminal scripts makes the environment artificial. There are tools available for capturing actual terminal input (for script creation), but even the output of these tools requires manipulation. Scripts rarely replicate all activity occurring on a system.

There are some standard benchmarks. In the HP3000 environment there is the "EDP" benchmark, which is described in the "HP3000 Performance Guide". And there are some industry standard benchmarks, such as the Whetstone benchmarks for measuring floating point performance. The extent to which these benchmarks are reflective of the performance of your applications depends upon the similarity of those applications to your environment.

The heart of a good benchmark is representative script creation. A terminal script simulates exact keystroke input; both in character representation, and in time between keystrokes and transmission to the host. At Hewlett-Packard we use an internal set of software programs called the Terminal Emulator and Performance Evaluator (TEPE) system. This software allows us to create scripts either manually (using a text editor or like process) or automatically. The automated technique uses a set of library routines that intercept terminal reads and logs them to disc.

The reason for complex script creation is to avoid bias due to memory buffering. For instance, if a the same transaction input executes repetitively, then the first execution will cause all

database validations to be brought into memory, and all subsequent executions will access the database validation entries from memory. Also, the memory and system requirements to run a single transaction can be significantly less than the requirements to run a large system of transactions.




The TEPE software, along with a TEPE configuration, also allows us to run benchmarks using those scripts, and analyze the benchmark results in terms of transaction rates and response times. A simulator system, called the TEPE system, simulates terminal input into the System Under Test (SUT) via cross connected RS-232 cables. The TEPE system records all SUT dialog, for later analysis.

Normally, a benchmark consists of multiple runs. Each run requires either adjustment of the TEPE script, or reconfiguration of the SUT. For proper comparison, the SUT database should be reloaded between runs. Obviously, the benchmark technique is costly and time consuming.

### Modelling.

Modelling represents a significantly lower cost alternative to benchmarking for predicting system performance. Modelling is technique of mathematically projecting performance based upon measured transaction volumes and system loading. Modelling offers the benefit over benchmarking in that a model can be based upon actual activity in a system as opposed to artificially created scripts.

Generally speaking, a model can only be applied to systems within the same computer system family. Within one family the code and I/O pathlengths for a given workload are constant. For example, a batch program on an HP3000 series 58 executes exactly the same instructions as does that same batch program on an HP3000 series 42. But the number or even types of instructions that would be executed on some other computer system would not be the same, since there would be different instruction sets, database management routines, and so forth.

	HP 3000	<b>SM02/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

It is possible to model applications on different computer families, but this requires a detailed analysis of each system. This type of model requires low level pathlength translations to create new workload definitions on the system to be modeled. This type of model is rarely created.

As with benchmarking, modelling standalone batch behavior is a simple task. Projections of standalone batch can be made by multiplying the portion of affected processing time (e.g. CPU busy percentage) by a value reflecting the speed of the new device (e.g. .5 for an HP3000 series 58 to HP3000 series 70 upgrade). But the essence of modelling is in predicting response times, transaction rates, and elapsed time for batch programs running in a multi terminal environment, possibly with mixed batch.

The modelling input data that we at Hewlett-Packard use is collected by the Measurement Interface and Data Collection Facility of the MPE operating system. This mechanism allows precise measurements of the work performed by all programs that were active during a measurement. Less sophisticated analyses can be made, however, using data from logfiles, customized measurements from intrinsics, and data extracted from tools such as the Online Performance Tools (OPT) or HPSnapshot reports.

Modelling is based upon the following simple assumption:

$$\text{Response Time} = \text{Service Time} + \text{Queueing Time}$$

Where *Service Time* is the time required in both CPU time, I/O time, and any software constraints (such as Image DBCB contention), and *Queueing Time* is time spent waiting for those resources.

*Service Time* is measurable. In an HP3000 system, the system logfiles (and also the MPE data collection facility) tell us the amount of CPU and disc I/O time required per program, which we divide by the number of terminal reads. This gives us the per transaction CPU and Disc I/O requirements for a given program or collection of programs (also called a workload).

*Queueing Time* is a function of device utilization. For example, if a disc drive is idle (other than by use of this singular workload), then the queueing time will be zero. If, however, the disc drive is in use eighty percent of the time, then the queueing time will be high, since the disc requests ahead of the current request must be honored before the current request can be issued.

Utilization of a device (which can be either the CPU or a disc drive) can be computed by the following formula:

$$\text{Utilization} = \text{Transaction Rate} * \text{Service Time}$$

Utilization is important in that it can be used to determine the queueing time element of response time. *Queue Length* is the

number of requests to be processed before the current transaction can be processed. The calculation of Queue Length can be found in most queueing theory textbooks.

$$\text{Queue Length} = \frac{\text{Utilization}}{1 - \text{Utilization}}$$

Queueing Time is the time for other requests, not including the current request, to complete. For systems with large transaction volumes, which would include most interactive terminal based systems, queueing time can be approximated by the following formula:

$$\text{Queueing Time} \sim \text{Queue Length} * \text{Service Time}$$

And, for systems with large transaction volumes, we can make a good approximation of response time as:

$$\text{Response Time} \sim \text{Service Time} * (1 + \text{Queue Length})$$

Now, for a simple example. Using the PROCTIME intrinsic of an HP3000 we can determine the CPU seconds for a transaction. For example sake, suppose that we determine that a transaction takes .5 seconds (or .0001 hours) of CPU time. And we anticipate 6000 transactions per hour. Then:

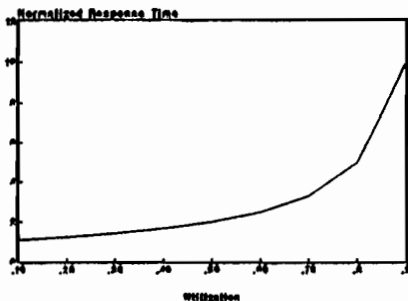
$$\text{Utilization} = .0001 * 6000 = .60 \quad \text{and}$$


$$\text{Queue Length} = .6 / (1 - .6) = 1.5 \quad \text{so}$$

$$\text{Response Time} \sim .5 * (1 + 1.5) = 1.25 \text{ seconds}$$

Simple!

With a few calculations it is possible to see the correlation between utilization and response time. The following diagram shows normalized values of response time versus utilization.



	HP 3000	SM02/7
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

This graph does not show a linear relationship. For example, between utilization values of .8 and .4, the normalized response time changes from 5 to 1.67. In other words, by halving the utilization, response time is reduced by almost two thirds.

It should be evident in the graph that somewhere past seventy-five to eighty percent utilization, that response time grows quite rapidly compared to utilization level. This range can be considered to be the maximum usable utilization for interactive processes. Some systems with large variations in transaction rates (and resulting peak levels) may consider smaller maximum utilization targets.

The queue length formula can also be applied in reverse. That is, we can determine the utilization for a given workload if we know the queue length. This technique allows us to isolate the needs of a critical workload from the entire system. This eliminates the effects of, for example, background batch. If we target a given processor utilization (for a critical workload) as a maximum, then we can derive how much additional capacity is available.

For instance, if the critical workload of a system spends as much time in the preempted state as it does in the CPU busy state, then we know that the queue length is one. Applying the queue length formula in reverse, we can derive that the CPU is fifty percent utilized for this workload. Using this measurement, we can assume that this system can tolerate fifty percent more activity by the critical workload. That is, a fifty percent increment in activity would raise the current fifty percent critical workload utilization to seventy-five percent utilization.


If this sounds too simple, well in many cases it is. Most systems run multiple workloads, and in many cases the workloads execute at different priorities. And in interactive database oriented applications there are considerations of multiple "server" devices; a system consists not only of a central processing unit, but also a set of disc drives. And oftentimes the desired calculations are based upon numbers of terminals, whose transaction rates can vary depending upon response times.

At Hewlett-Packard we use a computer program to perform the modelling of a complex system. And we deliver modelling results in the HPCaplan consulting package.

So, given these techniques, what can modelling produce? Our initial intention was to measure response time and transaction rates under different hardware configurations. Modelling gives us those capabilities. It also gives capacity projections for specific workloads. And since modelling gives ranges of response times, it also gives estimates of "reasonable" response times.

While modelling is normally used for projection of existing applications, it can also be used to model new applications. This can be done in an analogous technique to benchmarking: a



	HP 3000	<b>SM02/8</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

script can be made, and the results measured. For modelling purposes, human operators can be used to input data, and the script can be short duration. The purpose of the script, in this case, is simply to create a measurable system load. Therefore, the complete human operator load does not need to be put on the system; only enough load for the measurement to take place. Like benchmarking, this technique suffers in accuracy in that it is artificially created.

Modelling can also be used to predict performance with different workload mixes. That is, the work attributed to program A can be removed from workload calculations in a system containing programs A, B and C. Or program A from system X can be added to a system containing only programs B and C.

### In Conclusion.


Benchmarking and Modelling offer techniques for predicting computer system performance. These techniques allow us to predict external performance: response time, and transaction throughput.

The primary advantage of the benchmark technique is that it can be performed on any computer system. It can be used to measure delivered results as opposed to raw hardware speeds. Benchmark results are good for comparisons between system types.

Benchmark scripts, however, are artificial. Therefore, they may not be accurate projections of how an application will perform in use. Benchmarks are also quite costly.

Modelling offers a lower cost alternative for predicting system performance. Modelling offers the strong advantage that the activity that is used for the projections is collected from a live system. Modelling offers considerable flexibility. In addition to projecting response time, it can also predict system capacity, and performance with different workloads.

-----  
Larry Kemp is a Systems Consultant for Hewlett-Packard in the Bellevue, Washington (USA) office. Larry specializes in system performance and data management tools. He has a Master of Science (MS) degree in Computer Science from the University of Oregon (1975). Larry has worked at Hewlett-Packard since March of 1980.

	HP 3000	SM03/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## DISC PERFORMANCE - WHAT IS IT?

RICK ALDINGER  
HEWLETT-PACKARD COMPANY  
11413 CHINDEN BLVD.  
BOISE, IDAHO 83709

A typical HP 3000 environment measures disc performance and system throughput in transactions per hour or disc I/O's per second. Many MIS directors are interested in ways to get more out of their disc drives and CPU's. The purpose of this paper is to provide the audience with an overview of the factors involved in disc performance and some features available to you for maximizing performance.

The paper is divided into two separate segments. The first is a general discussion of the basics of disc performance as they relate solely to the disc drive. It explains the components of a disc transaction and the performance implications of each.

Segment Two discusses ways in which performance can be improved. This includes a discussion of cache and an explanation of RPS. It addresses ways your hardware configuration can affect performance as well as the role file management plays.

Any discussion of disc drives should begin with an understanding of the disc and its operation. In discussing disc performance it is necessary to first understand how a disc drive operates. Specifically, how transactions occur.

### Disc Transaction

A disc transaction is comprised of four components: disc controller overhead, physical seek, physical latency, and lastly, the actual transfer of information. Transfer time is the smallest component of the transaction, with controller overhead being next. The mechanical seek and latency comprise the largest part of the transaction time.

The following chart compares the transaction components of Hewlett Packard disc drives:

	controller	seek	latency	transfer	
7912	4.0	27.1	8.3	1.2	ms
7914	4.0	28.1	8.3	1.2	ms
7933	4.5	24.0	11.1	1.0	ms
7937	<1.0	20.5	8.3	1.0	ms
7945	10.1	30.0	8.3	2.0	ms
7958	3.0	29.0	8.3	1.2	ms

### Disc Controller

The disc controller provides the intelligence of a transaction electronically. It begins processing the transaction by:

- o Decoding the disc command sent by the host computer,
- o executing that command,
- o and finally, reporting the execution status back to the host.

The intelligence of the controller comes at the price of overhead in the disc transaction. However, experience has allowed us to make our controllers efficient in doing the greatest amount of work in the smallest amount of time.

### Seek Time

Once the controller has decoded the command, the disc must perform some mechanical functions to prepare for its execution. The drive must first find the desired disc location by moving its heads to the correct media track. The mechanical movement of the head to the desired track is defined as the seek.

The time to find the desired track varies depending upon its location on the media and the current position of the head. A more accurate estimate of seek time is the AVERAGE SEEK, or the time to do all possible seeks divided by the total number of seeks possible.

### Latency or Rotational Delay

Now that the drive has found the correct track it must now find the desired sector on that track. The media continues to rotate beneath the head as the track is searched. The time

required for one rotation of the disc is defined as the LATENCY time. While the media is rotating, the track is searched for the target sector. The rotation, like the seek, is mechanical.

This definition of latency is certainly a "worst case" time since the head may be considerably closer to the desired sector than one full rotation. A more accurate measure of rotational delay is the AVERAGE LATENCY. It is defined as the time to complete one half of a rotation.

**Transfer**

Once the head is positioned over the correct sector, it is time to transfer the data. TRANSFER is defined as the actual movement of data from the CPU to the disc (or vice versa).

HP defines AVERAGE TRANSFER as the average rate that data comes off the disc when reading an entire sector. Multiples of full sectors are always transferred in order to optimize performance. Partial sector transfers would require more bookkeeping and overhead.

**Transaction Time**

Each of the components of a disc transaction contribute to the total time involved in completing that transaction. The summation of the average time it takes to complete each component is a good measure of the total average time to perform a disc transaction.

The total AVERAGE TRANSACTION TIME for a particular disc product is defined as the sum of the average controller overhead for the product, plus the product's average seek time and rotational delay, plus the average time to transfer one kbyte of data to the product. The total average transaction time is specific to the product in question and, as we have defined it, does not take into account individual host system attributes.

The following figures are the various transaction times for Hewlett Packard disc drives:

HP 7912	40.6ms
HP 7914	41.6ms
HP 7933	39.6ms
HP 7937	30.8ms
HP 7945	50.4ms
HP 7958	41.5ms

### Performance Metric

Hewlett-Packard uses the metric of I/O PER SECOND to measure disc performance. I/O per second is defined as the maximum number of disc transactions per second that a specific drive can perform at a transfer size of 1 kbyte. This measure is calculated by taking the inverse of the total average transaction time just described. It measures raw disc performance and does not take into account any system specifics. Actual performance will vary with system and application. I/O per second is a Hewlett Packard measurement and not an industry standard.

Let's go ahead and convert the transaction time of an HP 7958 into the measure of I/Os per second. We already learned that it takes the 7958 41.5 ms to transfer 1 kbyte of data. If we inverse our measure we can learn how many 1 kbyte data chunks can be transferred per unit of time. By converting milliseconds to seconds we have a measure of the number of kilobytes, or I/Os we can transfer in one second.

EXAMPLE:  $1\text{kbyte}/41.5\text{ms} * 1000 = 1\text{kbyte}/.0415 = 24.1 \text{ I/O per sec}$


The following figures are the I/O per second measurement for some other Hewlett Packard disc products. Again, this measure is for relative disc performance only and does not take into account system overhead. Actual performance will vary with system and application.

7912	24.5	I/O per second
7914	24.0	I/O per second
7933	25.3	I/O per second
7937	32.5	I/O per second
7945	20.0	I/O per second
7958	24.1	I/O per second

Well that wasn't so bad, was it? Now that we are aware of the drive's performance, let's focus on the options you have to improve disc drive efficiency. Please keep in mind the items in the following paragraphs are very system and application dependent. These are guidelines for you to use to help in performance tuning your system.

### Disc Controller Cache

Disc controller cache is a method for improving performance. It is a RAM based storage area resident in the disc controller that provides high speed access to data. Frequently used data (directories, for example) are stored in the cache area, rather than on the disc media. For every cache access, a seek

	HP 3000	<b>SM03/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

and latency are eliminated. The greater the "hit ratio" to the cache, the greater the performance improvement!


Cache is currently implemented only on the HP 7933, HP 7935, HP 7936 and HP 7937. We do plan to implement it on future high end disc products. It is supported on both the HP 3000 and HP 1000. Support for the HP 9000 is under investigation.

If we refer to our original transaction, we see that actual disc performance can be improved by decreasing the time it takes to perform even one component of the transaction. Little benefit is seen from improving transfer rates since they are only a very small component of the overall transaction. Controller efficiency has been fine tuned to a degree that shaves away most excess overhead. What about seek and latency, the largest components of the transaction? Better than reducing seek and latency times, controller cache can often times eliminate the need to perform either operation! MPE CACHE, resident in the CPU, eliminates disc controller overhead when the desired information can be accessed from the cache. The only component is the transfer of data out of CPU memory.

When is Disc controller cache better than MPE disc cache? A lightly loaded, non-cached system (less than 75% CPU utilization) will benefit from either MPE disc cache or controller cache. This type of environment is likely to be more I/O bound than CPU bound. Both caching schemes greatly reduce the I/O bottleneck. MPE disc cache will have a slight advantage over controller cache because there is no controller overhead involved when reading directly from main memory.

As the CPU load increases to a moderate level (75% to 90% utilization), the throughput of a system with MPE disc cache is impacted. The management of MPE disc cache must now compete for fewer available CPU cycles. MPE disc cache will continue to be faster than a non-cached system at this stage, but the potential for controller cache to become more effective greatly increases. Controller cache begins to provide the capability of leveling out CPU peaks.

When a system reaches the stage where CPU load is heavy (90% and above), the impact of MPE disc cache on system throughput can become negative. In extreme situations, the system may actually perform more efficiently with MPE disc cache turned off. In this environment controller cache provides a noticeable benefit, especially when MPE disc cache logical read/write ratios and read hit percentages are high. This indicates that a good deal of I/O is being eliminated and that CPU cycles and memory used for managing cache can be freed for other activities.

	HP 3000	<b>SM03/6</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Controller cache will continue to significantly outperform a non-cached system until the CPU load is increased to the point where the system is so CPU bound that I/O is no longer a factor.

### Rotation Position Sensing

Use of Rotation Position Sensing(RPS) is another means by which performance may be improved. RPS is a disc feature designed to minimize non-productive use of the channel while waiting for the disc to locate the area at which a transfer will begin. There is a window of time after the drive receives a command and before it finds its target sector that is generally wasted. RPS allows the channel to accept another command during that window, thus utilizing the channel better.

As might be expected, RPS provides little benefit to single drive configurations, since the channel is not the bottleneck here. But, in multiple drive, multiple process configurations, RPS can help to relieve channel contention. RPS is supported on HP 791X and HP 793X disc drives and only on the HP 3000 systems.

For instance, when a request is generated(like a read or write) that requires a disc I/O, the CPU sends a command across the channel to the disc drive. There is a window of time, after the drive receives the command and before it finds the target address on the media, that the channel is not released and cannot be used for another request. No other drives on the channel can be accessed during this window.

RPS allows the channel to be used during this window for access to the other drives on the same channel. As soon as a drive receives the command, it disconnects from the channel. During this time, other channel activity can occur. When the target address in our original transaction is found, the drive reconnects to the channel. If the channel is busy at the time, the data is buffered until the channel is free.

### System Configuration

The physical location and configuration of the disc drives has a big impact on performance. Questions like, "Where do I put my system disc?", "Do each of the discs need separate interfaces?", and "When should I use multiple drives over a single, larger one?" can all be answered to optimize performance.

Please keep in mind, the optimal solution depends on the CPU in use, the number of users on the system, the application, etc. The following guidelines are conceptual and may not apply to all systems.

The location of your system disc has a big impact on performance. The optimum configuration places your system disc on a dedicated interface. With your system disc here it won't have to compete with other drives for channel activity. Since the system accesses this disc most often, a performance improvement can be realized.

As implied, channel contention can have a very negative impact on performance. Systems with many users and multiple processes accessing multiple drives can generally get relief by putting their drives on separate, dedicated interfaces. For systems with many drives this may not be economically feasible or it may exceed the maximum number of interfaces the system will permit. In these cases RPS or cache may be viable alternatives.


With the multitude of disc drive capacities to select from, it often becomes unclear which combination of drives optimizes performance. Of specific interest is the question of two smaller drives versus one larger one. The answer is very application dependent. For a system that can keep both drives busy, two drives on separate interfaces is generally the best answer. The system can then access both concurrently, increasing overall performance. On the other hand, a single faster drive is the better solution for a very localized system with little multiple processing occurring.

For instance, let's say you have a Series 68 currently configured with one IMB (inter modular bus) and two high speed GICs (general interface channels). On one GIC you have four HP 7937s, and on the other you have an HP 7978B. You would like to add an HP 7937 and an HP 2680 printer. What can you do to help increase performance?

A heavily used HP 7978 tape drive and a HP 2680 printer should have dedicated GICs. In order to do this another IMB must be added. The new IMB can then accommodate two new GICs, one each for the tape drive and printer. The HP 7937 disc drives can then be spread over the two remaining GICs on the first IMB. You could put two HP 7937 disc drives on one GIC and three HP 7937 disc drives on the remaining one.

If the tape drive is not heavily used it could share a GIC with the printer. This would free up a GIC and allow you to spread the discs out even more. This configuration would



	HP 3000	<b>SM03/8</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

allow your system disc to reside on a dedicated interface, and two drives on each of the remaining two GICS.

### **File Management**

Where you physically locate your files on the discs also impacts performance. A good general rule is to spread your system files, virtual memory and other user files (including databases and spooling operations) evenly among your discs. Keeping your system files and your virtual memory on separate discs is most beneficial. By spreading your files around, the system will experience less contention in accessing the desired areas.


The biggest gain will be seen by spreading data sets and other MPE files among the various disc drives. It is also helpful in performing a reload every quarter. This helps eliminate much fragmentation that usually occurs on a heavily loaded system. When you perform the reload it is best to do an accounts reload and then restore the most heavily used files at the front of the disc. The least used files should be loaded last. This provides some benefit because the heavily used files are closer to the system directory. This helps reduce some of the disc's mechanical functions.

Overall, there are several options the performance conscious user has beyond raw disc performance. They include:

- o DISC CONTROLLER CACHE as a means of reducing mechanical seeks and latencies,
- o ROTATION POSITION SENSING to relieve channel contention,
- o SYSTEM CONFIGURATION for the most efficient use of resources, and
- o FILE MANAGEMENT to optimize throughput.

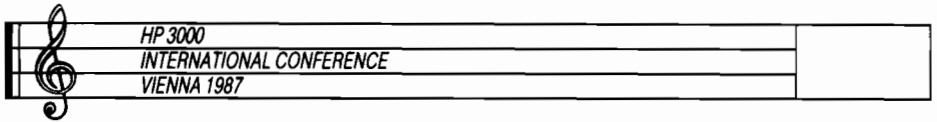
Keep in mind it may take one or all of the many items discussed to improve your performance. Each option can affect a system differently because of the applications being run and the amount of users on the system.

Do not become disenchanted if one of the options does not work. Take time to experiment with your system and the items we discussed and ask your SE and CE for help. They are an excellent resource. HP OPT/3000, HP SNAPSHOT and HP TREND are good tools in helping determine file placement, usage etc. When you take a look at your system the results may be

	HP 3000	SM03/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

different each time. What might have worked one time may not produce the same high benefit the next time. However, remember that you are trying to level out the "peaks" and "valleys" on your system, providing your user with a much more balanced system.

Last but not least, use good **COMMON SENSE**.



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

## Performance Measurement for Capacity Planning

*The Key to Information*  
Interex 3000 European Conference  
Vienna, Austria

Tim Twietmeyer  
Hewlett-Packard Company  
Performance Technology Center  
Roseville, California, USA



---

*This paper is a logical extension of papers published by Paul Primmer in the proceedings for the Interex Conference in Amsterdam, 1985, and Interact magazine in November, 1985. In Paul's papers, he details the development and results of the System Performance and Evaluation Project (SPEP) begun at HP Labs in 1983. This paper will review the SPEP and will discuss the new developments which have evolved from the SPEP project.*

### Introduction

What began as a small project to collect customer data to aid in the development of future systems and products has since become a major contributor to performance measurement technology for Hewlett-Packard and its customers. The System Performance Evaluation Project at HP Labs was chartered with the task of collecting performance data from customer's machines for making trade-off decisions on the design of future systems, characterizing customer workloads, and refining HP product usage. From this project, a more complete package for evaluating system performance has been developed.

SPEP, the System Performance Evaluation Package, has developed into a comprehensive performance measurement, evaluation, and capacity planning tool. There are five main parts to the SPEP:

- **Collection** - The collection tools probe the system under evaluation and captures the data needed to complete the performance evaluation.
- **Reduction** - The reduction tools take the raw data collected from the measured system and produce reports and additional files for review by the performance specialist.
- **Summary** - These tools summarize the measured data from the reduction section and stores them with other collected sites for later multiple machine studies.
- **Capacity Planning Workbench** - This allows the performance specialist to review and manipulate system workloads for use by the analytic modeling tool.
- **Analytic Modeling** - This step takes the workloads defined by the workbench, validates the values produced by the analytic modeling tool, then predicts throughput, utilization, and response time.

It's the SPEP that the HP field performance specialist uses to deliver the HPSNAPSHOT and HPCAPLAN consulting products. With the tools of the package and the knowledge and experience of the performance

specialist, system bottlenecks are identified and capacity planning questions are answered. Now, let's take a more detailed look at the purpose and mechanics of each SPEP section.

### Collection Phase

The collection process is responsible for acquiring all the pertinent data a performance specialist needs to perform their analysis. To be successful the collection process must meet several objectives. Let's look at these objectives and see how the SPEP satisfies them.

*The process must be simple to install and complete for the operators of the system under analysis.*

To facilitate a simple procedure for installation, SPEP uses a specially formatted tape to place the collection software on the system under analysis. A variation of the STORE tape format is used:

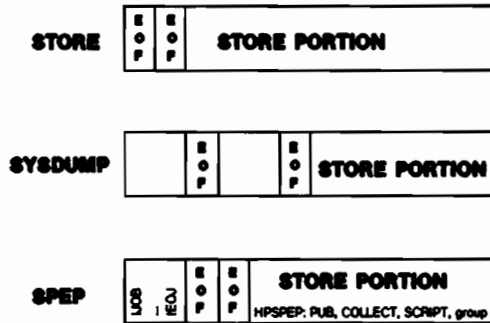


Figure 1. SPEP Tape Format

A special PRBEBOOT file is written to the front of the tape and the collection software is then appended after two EOF markers. This allows the operator to mount the tape, enter a tape file equation:

```
:FILE T:DEV=TAPE
```

and stream the file from the front of the tape.

```
$STREAM *T
```

This will cause the PRBEBOOT jobstream to execute. It will create the accounting structure for the collection tools and then execute the RESTORE program to install the tools. The collection begins after an operator reply to acknowledge the company name.

When the collection tools have completed, STORE is used to write back to the original tape the log files that were produced by the tools. This tape is returned to the originating HP site for reduction and analysis.

Using this method, the only required intervention by the system operator is responding to three tape requests. This makes the collection appear very simple when actually it is fairly complex. A discussion of the collection tools themselves can be found later in this section.

*The collection tools should not significantly impact the normal processing of the system.*

The second, and perhaps the most important objective is to collect the data without perturbing the system. The collection process was originally designed to run all collection tools simultaneously. This consumed 17 percent of the CPU and was considered too much. The collection was then changed to use two 1-hour phases, running SAMPLER (APS/3000) in the first hour and the other collection tools in the second. This reduced the overhead induced by the collection tools but it introduced the problem of SAMPLER recording a different window of measurements than those captured in the other log files. The current process runs all the collection tools in one collection interval, usually a single hour. By tuning the sampling interval in the tools to balance the amount of overhead and the amount of data needed to accurately represent the system, the overhead was reduced to 5%-7%. This is considered acceptable for the accuracy and volume of data being collected.

*The process should not monopolize system resources during the collection.*

Along with keeping the overhead of the collection low, SPEP also attempts to keep the amount of physical resources consumed low. To avoid monopolizing a tape drive for the entire collection interval, the package logs its data to disc. With this decision comes the problem of finding adequate disc space.

To avoid having to abort the collection from a lack of free space, the collection runs a program to verify that free space is available before any collecting begins. This program, BEGIN, checks for free space depending on the system type and the length of collection requested. An average of 50,000 sectors is anticipated. If the free space is inadequate, the collection can be terminated by the operator.

*The process must be flexible so the performance specialist can modify the collection duration.*

Depending on the size of the system and the application mix that's to be measured, the collection interval may need to last anywhere from fifteen minutes to four hours. Most SPEP collections are built to run for a one hour interval. One hour is used because a shorter period may not capture a large enough sample and a longer period may cause problems by consuming too much disc space.

When the probe tape is being created the collection duration is determined. The duration for the collection is prompted for and a corresponding script file is then chosen so that all collection programs run for this fixed period of time.

*All data needed to complete the analysis must be collected in one pass.*

SPEP is designed to be an automated package which can collect a large amount of performance data but require little human intervention to execute. The situation we are trying to eliminate by using SPEP is that of a performance specialist spending days manually executing different collection tools trying to ensure that they have isolated the area they need to examine. With SPEP, the specialists can send the user a collection tape with instructions, have the collection executed, and the tape returned with the collected data. They can then complete their analysis without spending several hours working on the system under study.

### Collection Tools

There are several different tools used by SPEP to complete its collection process. Some of these tools are supported HP products and others were written by HP engineers to support SPEP or to provide a tool that wasn't offered by MPE. The collection tools and their corresponding output files are illustrated below:

### SYSTEM PERFORMANCE EVALUATION PACKAGE

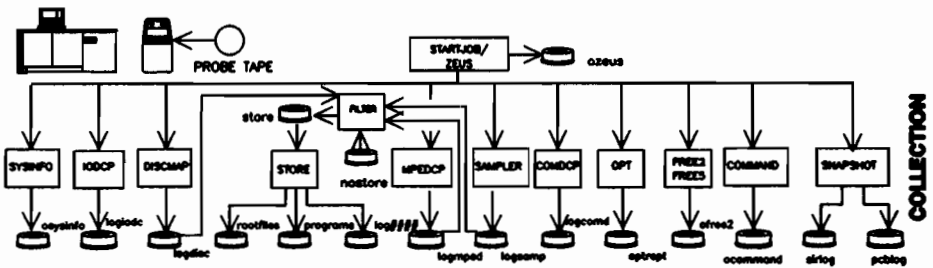


Figure 2. SPEP Collection Tools

The following illustration shows the collection sequence from start to finish:

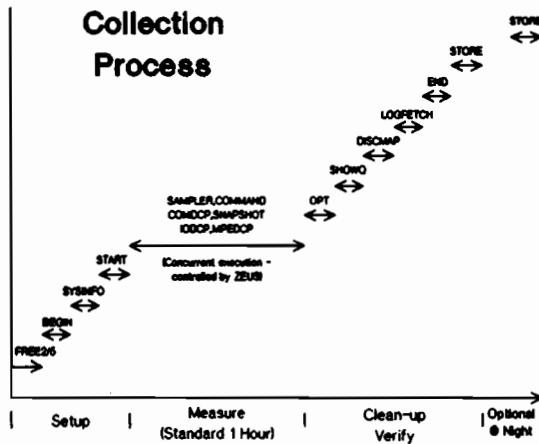
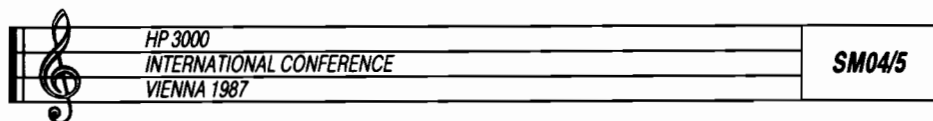


Figure 3. SPEP Collection Sequence



Performance Measurement for Capacity Planning  
Interex 1987, Vienna, Austria

After the system operator restores the files from the probe tape, the setup portion of the collection begins. FREE2/FREE5 produces the OFREE file which contains a listing of the free disc space and disc fragmentation. The BEGIN program then checks for sufficient free space, software update levels, and table sizes which may cause the collection to abort if it continued. SYSINFO then creates the OSYSINFO file which is a listing of the system configuration. The START program then prompts the operator for the company name and the measurement section of the collection begins.

ZEUS controls the execution of the measurement section which is usually specified to run for one hour. ZEUS uses process handling to spawn the programs so they can run concurrently. SAMPLER logs records to the LOGSAMP file every 50 milliseconds. SAMPLER monitors time spent in software modules. COMMAND executes a SHOWCACHE command and writes it to the OCOMMAND file once every 60 seconds. COMDCP executes a SHOWCOM command once every minute for all configured data communication devices (INP, SSLC, LANIC, and HSI). SNAPSHOT awakes every 10 milliseconds and copies the entire PCB table and SIR table to a disc file. IODCP turns on the I/O trace facility in the measurement interface and logs a 20 word block of I/O information for every I/O. Lastly, MPEDCP is started to collect global and process level information at one minute intervals.

The clean-up section begins with OPT capturing the table sizes. SHOWQ is then used to record the MPE queue configuration. DISCMAP scans the directory and captures the 128-word file label for every disc file on the system. Logfetch creates an indirect STORE file so MPE and Network log files can be written to tape. END verifies that all log files were created during the collection tools and that each of these log files contains records for the duration specified by ZEUS. Finally, STORE writes all the log and listing files to tape.

The optional section shown in Figure 3 stores program and root files that were flagged during the collection.

The entire collection process, from the building of the accounting structure to the store of the files to tape, will complete in two hours. The collection tape is then returned to the originating HP office and reduction of the data can begin.

## Reduction Phase

It's the function of the reduction software to process the raw data collected from the system under study. This process produces the standard performance report and creates the files needed for the summary and capacity planning phases.



As with the collection process, the reduction phase uses both HP supported products and special programs written to support the SPEP package. The illustration below shows the flow of data from the collected log files to the creation of the performance report:

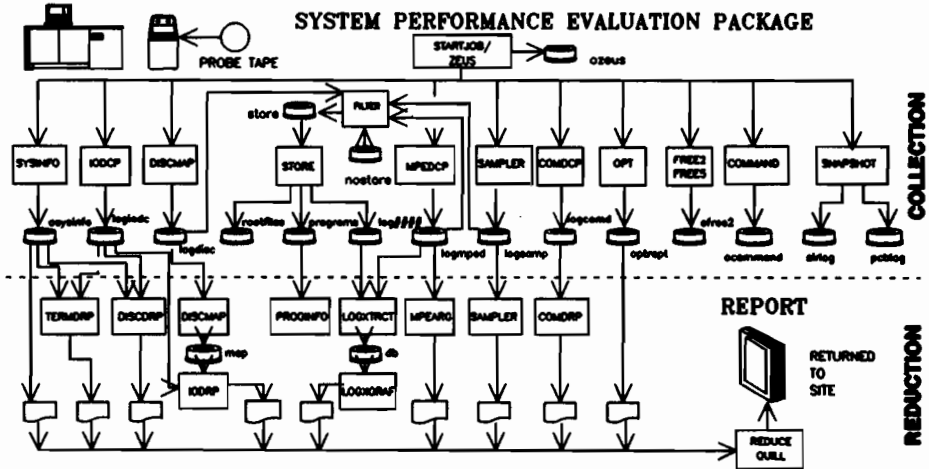


Figure 4. SPEP Collection and Reporting

Each of the tools creates a different section of the final standard performance report which is used by the performance specialist to diagnose system problems. The execution of the tools and the creation of the report is controlled by a single jobstream. The report sections are created by the following tools:

**TERMDRP** - The system configuration captured by SYSINFO is used to identify terminal devices. The I/O trace from IODCP is then searched to reconstruct the transactions for each terminal. These transactions are used to determine think times, response times, and average data transfer per transaction. These are presented as histograms in the report. A numerical summary is also created that reports terminal reads, writes, and control functions per terminal and per transaction.

**DISCDRP** - This step also uses the system configuration captured by SYSINFO. It identifies disc drives and then uses the I/O trace file to calculate disc addresses accessed, seek distances, and data transfer sizes. This data is presented as histograms in the report.

**DISCMAP** - The file labels captured in LOGDISC are used to build a KSAM file with the file's extent addresses as the keys. This file is used as a directory of file names ordered by extent addresses. IODRP then replays the I/O's and matches the disc addresses of the I/O file with the KSAM directory. The resulting report shows logical and physical I/O by file name and file type.

**LOGXTRCT/LOGXGRAF** - These tools produce the graphical section of the report. LOGXTRCT creates an IMAGE data base of the data found in LOGMPED. From the data base, LOGXGRAF creates line, pie, and bar charts showing global system states, disc queue lengths and throughput, and workload specific information.

**MPEARQ** - This tool produces a numerical report of the global system states and process level statistics.

**SAMPLER** - The report is a histogram showing the time spent in user code versus system code, and the modules in which the system was spending its time.

**COMDRP** - This tool reduces the LOGCOMD file then creates a graph and numerical data showing line utilization, error rate, retransmissions, and timeouts for each active INP, SSLC, LANIC, or HSL.

The SYSINFO listing and a listing of the table sizes from OPT are appended to the report. Other files not mentioned are either used later in the summary phase, or collected for reference if needed.

Each section of the report is output to a disc file. The report is assembled using the document preprocessing software named QUILL. One large TDP file is built and output to a laser printer. The HP2688 and HP2680 printers are supported. Production of the report takes approximately one hour. A portion of this time is spent creating files that are used in the summarization phase.

The performance specialist can use the report to identify disc bottlenecks, CPU saturation, memory pressure, and other system problems. If more detailed information is required to isolate a problem, several tools offer user interfaces to interrogate the log file online. In most cases, the online use of the tools is used to isolate a certain program or view a certain window of the collection period.

## Summary Phase

The summary phase of SPEP was designed to meet the most important objective - accumulating a large amount of customer performance data to analyze customer workloads and create patterns of customer usage. To meet this objective the summary data base (SUMDB) was developed. It is an IMAGE data base that stores summary records of each collection. The loading of the site into the SUMDB follows the completion of the reduction phase. A schematic of the SUMDB appears below:

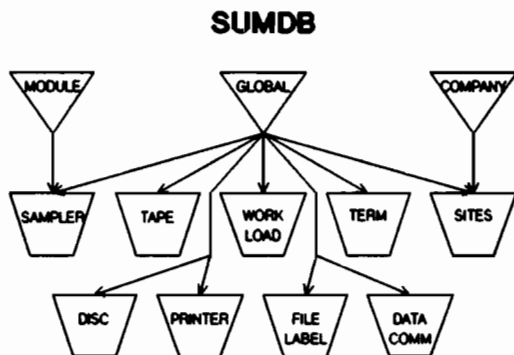



Figure 5. Summary Data Base

**GLOBAL** - Global retains the system configuration and summary information about the system and collection. Each site includes system type, number of terminals configured, date of collection, operating system level, site identifier, caching totals, and memory manager activity summaries.

	HP 3000	<b>SM04/8</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Performance Measurement for Capacity Planning  
Interex 1987, Vienna, Austria

This data is used to find the average number of terminals per system type, average memory size by system, and to compare memory and caching access patterns.

**TERM** - This set contains a record for each active terminal. Active is defined as one read/write pair during the collection period. Each record contains information pertaining to number of reads, writes, transactions, and the amount of data transferred per I/O.

This information is useful in comparing read-to-write ratios of systems and the average amount of data transferred per transaction per terminal.

**DISC** - Each disc drive is represented by one record. This set tracks disc queue lengths, seek distances, service times, and reads and writes by type. Model and LDEV number are also tracked.

This data is used to find average drives per system, average service times by drive type, and average queue lengths. Several curious questions can be answered by examining this data set, such as - Do systems tend to queue more on certain LDEVs?, How many drives are cached per system?, What percentage of I/O's directed to a drive are code reads, data reads, data writes, etc.?

**TAPE, PRINTER** - These sets track read, write and transfer per I/O information per device.

**DATA COMM** - Each data communications device is represented in this set. This includes the software product which uses the device, reads and write counts, transfer sizes, line speed, and time busy.

**SAMPLER** - Contains information regarding the activity in the various operating system modules for each site.

**FILELABELS** - By far the largest data set in the data base, this set stores an entry for every permanent file found on the collected system. Each entry includes access dates, record and block sizes, file type, number of reads and writes to the file, extent sizes and the number in use, and the number of records in the file.

Studies on this file can take hours of CPU time. Some questions that can be answered are - What are the average number of files per system, What is the average file size, What percentage of files are program file, etc.

**COMPANY** - Tracks the company name and market cell. Market cells defines the primary use of the computer at that company, i.e. Office Automation, Manufacturing, Financial.

**SITES** - This set retains customer information so the data can be traced back to the customer and machine it was collected from.

**WORKLOAD** - Without a doubt, the most important data set. This set represents a major effort in the summary and capacity planning phases of SPEP. It is from this set that the primary objective of the SPEP package is met.



Performance Measurement for Capacity Planning  
Interex 1987, Vienna, Austria

The program WKLOAD is used to load the WORKLOAD data set and uses several different log files from the collection. The flow of data through the WKLOAD program is shown below:

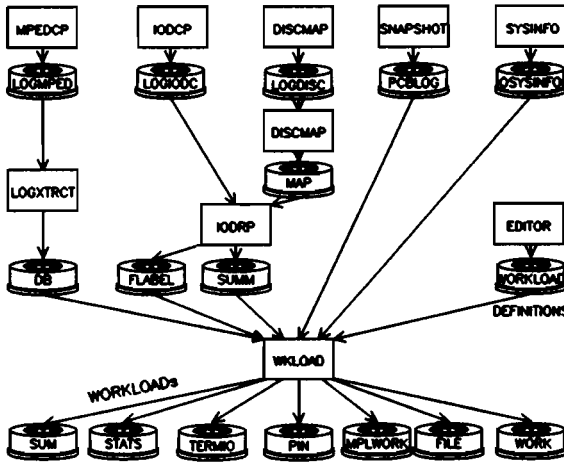


Figure 6. WKLOAD Data Flow

WKLOAD reads DB to find process level statistics which were collected by the measurement interface of MPE. FLABEL and SUMM are used to calculate I/O's by type (KSAM, IMAGE, MPE, Memory Manager, and Directory), think times, response time, and transaction counts. PCBLOG is read to calculate the MPL (Multiprogramming level) which is a metric that defines software constraints for a workload. OSYSINFO contains the system configuration listing and WORKLOAD is the workload definition file.

The WORKLOAD definition file defines the program names which make up workloads. Each workload defined in WORKLOAD is identified by a unique name, is assigned a category, language, and a set of filenames that will be bundled to create the workload. An example of a workload definition is:

```

WORKLOAD=TDP/3000
CATEGORY=OFFICE TEXT PROC.
LANGUAGE=SPL
TDP.PUB.SYS
TDPSP1.PUB.SYS
TDPSP2.PUB.SYS
SCRIBE.PUB.SYS

```

Each workload is also differentiated by whether it was run as a job, session, or as part of the operating system, and also by the queue in which it was executed. If a program file is not defined in any workload definitions, the file name is used as the workload name. Of these self-defining workloads, only those which consume over ten percent of the CPU or disc is isolated in its own workload. The remaining self-defining workloads (<10%) are bundled into a single workload defined as \*OTHER.

At the completion of WKLOAD the WORKLOAD data set in SUMDB might contain the following workloads for a site:

Workload	Type	Queue
FCOPY/3000	J	C
FCOPY/3000	S	C
HPDESK	J	D
HPDESK	S	C
HPWORD	S	C
*OTHER	S	C
SYSTEM	O	L
SYSTEM	S	C
SPOOLER	O	L
TDP	S	C
AP.PROG.FIN	S	C
PAY.PROG.FIN	S	C

The remaining information that would be found in an entry in WORKLOAD includes: reads and writes per I/O type, number of transactions, CPU time and instructions, active terminals, think time, response time, code faults, number of active terminals, think time, response time, number of code and data segment faults, and percentage of disc I/O to each disc LDEV.

It is obvious from the above listed data that a wealth of information can be gleaned from the SUMDB. A master version of the SUMDB is maintained by the Performance Technology Center in Roseville, California. This version of the data base is used to characterize HP products, create benchmarks, and track new product performance. Currently, the master SUMDB has over 350 collections and is updated daily with new data. The data base is distributed to HP divisions for use on the following projects:

- Computer Systems Division to characterize large Series 68 and 70 systems.
- Office Systems Division to characterize office systems software.
- Information Technology Group to create HP utilities benchmarks and to isolate certain customer workloads. The type of workloads being examined are intense IMAGE applications, heavy batch processing, manufacturing, office automation, and program development.
- HP performance specialists to use as input for the analytic modeling tool. Standard Workloads for several HP products have been developed and are used by performance specialists when delivering the HPCAPLAN product.
- The HP Software Migration Center for application migration analysis.

### Capacity Planning Workbench Phase

With the data that was stored in SUMDB, it became clear that by using certain mathematical equations, performance predictions could be calculated. This spawned the development of KASANDRA<sup>1</sup>, the analytic modeling tool. With this tool, a performance specialist could answer different "what if" questions concerning system utilization, response time, and throughput. The tool was completed but manual extraction of workload data from the SUMDB was tedious and time consuming. A workbench tool was developed to eliminate this problem.

The Capacity Planning Workbench (CAPWORK) provides an environment which allows the user to analyze the measurements from the customer collection, regroup or isolate workload groupings, and generate modeling information for input to KASANDRA. A data base (CAPLAN) is built by the CAPLOAD process and it's with this data base that CAPWORK interacts when modifying workloads.

Below is a schematic of the CAPLOAD process, including KASANDRA and standard workloads:

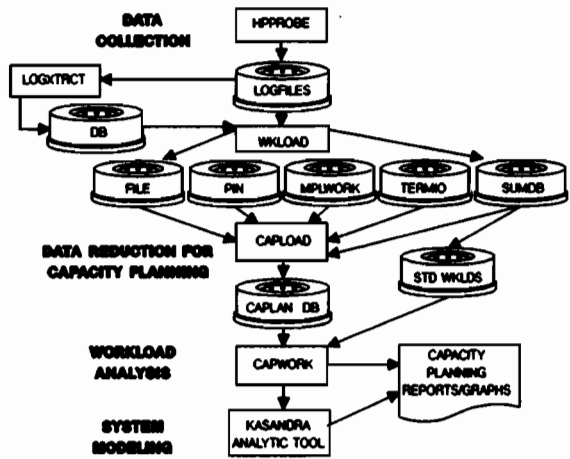


Figure 7. Capacity Planning Environment

When the build of the CAPLAN data base is complete, the Capacity Planning Report (CAPREPT) is output. This report shows the groupings of workloads produced by CAPLOAD and their associated statistics. It's the CAPREPT which the performance specialist uses for reference when modifying the workload groupings. This report can be generated after each regrouping of the workload structure. Each report includes: site configuration, modeling parameters for KASANDRA, interactive and wait state detail, service utilization, device utilization, workload composition by program name, file access by workload, file access by filename, and a listing of the workload definition file (WORKLOAD definitions from Figure 6).

<sup>1</sup>Named from the Greek mythological character Cassandra (often spelled CASSANDRA), the daughter of Priam and Hecuba. Apollo gave her the gift of prophecy to win her love. When she wouldn't yield to his desires, Apollo cursed her to never be believed. It's no wonder her warnings against the dangers threatening Troy fell on deaf ears. The spelling, KASANDRA, was used to satisfy the MPE file system.

The CAPLAN data base organizes the workloads into a tree structure. The highest level of the tree is considered the system level and this is subdivided by run environment, session, batch job, or operating system. The level under the run environment represents categories as described by the WORKLOAD definition file. The program level links to the category level and the process level reports to the program level. The entire tree consists of six levels of detail, the system level, run environment level, category level, workload level, program level, and process level.

The following illustration is an example of the workload tree:

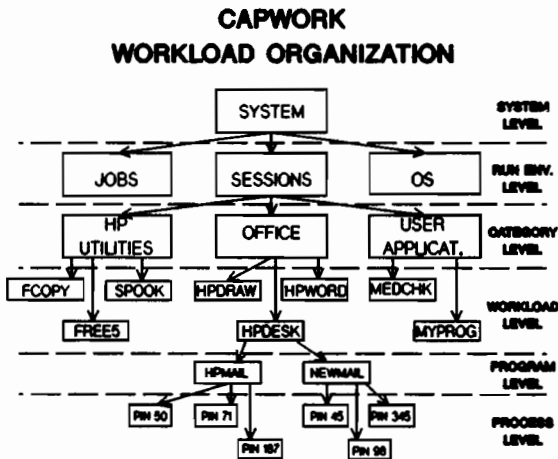


Figure 8. Workload Tree

From this structure the CAPWORK user is free to reorganize the tree to isolate certain applications such that, when KASANDRA is invoked, the workload is modeled as a single application and not bundled within its category. For example, in the above figure the following workloads would be passed by CAPWORK to KASANDRA; JOBS, OS, HP UTILITIES, USER APPLICATIONS, and OFFICE. It is rare to find JOBS or OS broken to lower levels since its seldom that JOBS or OS are of concern to the specialist. Most often it is online applications that require attention and this is why SESSIONS has been subdivided to the CATEGORY level.

Let's assume the objective of this capacity planning process is to determine the impact of ten new HPWORD users, twenty new HPDESK users, and evaluate the effect of moving the MEDCHK application to another system. The specialist would need to modify the CAPWORK tree structure to isolate HPWORD, HPDESK, and MEDCHK. This is done by summarizing the levels below these entities and separating them as a category. When this is complete, the workloads passed to KASANDRA would be; JOBS, OS, HP UTILITIES, USER APPLICATIONS (without MEDCHK), OFFICE (without HPDESK and HPWORD), MEDCHK, HPWORD, and HPDESK. With this organization the user can answer "what if" questions with each workload.



Performance Measurement for Capacity Planning  
Interex 1987, Vienna, Austria

When the user is satisfied with the workload structure which is defined, the KASANDRA analytic modeling tool is invoked. The process is then iterative, moving between CAPWORK and KASANDRA evaluating different system configurations, processors, and applications.

### Analytic Modeling Phase

A model is an abstraction of a system and represents an attempt to distill, from a mass of details that represent a system, those aspects that are essential to a system's behavior. In the case of KASANDRA, a queuing network model is used to represent the system. This particular method is an approach to computer modeling in which a system is represented by a network of queues which are evaluated analytically. The network of queues is a collection of service centers, which represent system resources, and customers, which represent users or transactions. KASANDRA solves a set of equations induced by the network of queues and its parameters.

The reasoning behind this choice of modeling technique is it achieves a favorable balance between accuracy and efficiency. Accuracy is expected to be within 5%-10% for utilizations and throughput, and within 10%-30% for response times. Efficiency is realized since defining, parameterizing, and evaluation of these models comes at a relatively low cost. The relative increase in cost of improving accuracy obtained by using other methods significantly outweigh the benefits for a wide variety of applications.

The specific equations used by KASANDRA to model systems are explained in *Quantitative System Performance, Computer System Analysis Using Queuing Network Models*, Edward D. Lazowska, John Zahorhan, G. Scott Graham, Kenneth C. Secik, Prentice-Hall, 1984. This book was used as a reference while developing KASANDRA. The particular algorithms are defined as Mean Value Analysis (MVA). No attempt will be made by this paper to explain MVA, only the results that it produces.

An MPE file of workload model inputs is passed from CAPWORK to KASANDRA. A sample of workload attributes appears below:

Title MODEL FOR KASANDRA

SYSTEM 70

Server 1; Time=0.0316; Title=DEV1  
Server 2; Time=0.0362; Title=DEV2  
Server 3; Time=0.0345; Title=DEV3

Create OS

CPU= 582.25,70  
Disc= 28769.609  
Drive= 1, 41.90  
Drive= 2, 33.35  
Drive= 3, 24.75  
MPL= 0.407  
PRI=1  
Type=Transaction  
Thruput= 0.000278

Create JOBS

CPU= 818.788,70  
Disc= 40515.886  
Drive= 1, 33.33  
Drive= 2, 33.33  
Drive= 3, 33.34  
MPL= 5.496  
PRI=3  
Type=Transaction  
Thruput= 0.000278



Performance Measurement for Capacity Planning  
Interex 1987, Vienna, Austria

Create OFFICE TEXT. PROC		Create USER	
CPU=	0.403,70	CPU=	0.170,70
Disc=	14.400	Disc=	3.684
Drive=1,	24.00	Drive= 1,	33.00
Drive=2,	36.00	Drive= 2,	33.00
Drive=3,	40.00	Drive= 3,	34.00
MPL=	1.124	Pri=2	
Pri=2		Type=Terminal	
Type=Terminal		Term=	13.45
Term=	5.62	Think=	12.2
Think=	9.677		


This particular example defines four workloads. Each workload is defined by the CREATE command followed by the workload attributes. CPU is defined as the number of CPU seconds per transaction and the '70' represents the processor. DISC defines the number of disc I/O's per transaction. The DRIVE command defines the percentage of I/O's that were directed to a particular drive. PRI assigns a priority level corresponding to the queue in which the workload executed. MPL<sup>2</sup> defines the degree of software constraints for the workload. TYPE defines the type of transaction. For batch jobs and operating system the TRANSACTION type is used with the THRUPUT command. This is used to saturate a fixed amount of resources by the workload. For interactive workloads we use the type defined as TERMINAL, the THINK command which is a measured think time for a transaction, and the TERM command which defines the number of terminals that should be modeled for this workload. SERVER defines the disc devices and the average service times measured during the collection.

KASANDRA will take the workloads defined and predict the response time, utilization of devices, and throughput. These values are compared to those output by CAPWORK in the CAPREPT. If the values compare satisfactorily, the user is ready to predict new values for throughput, utilization, and response by changing the system type, number of disc drives, number of terminals that run a particular workload, or perhaps disc service times. If the values calculated by the model are not in the acceptable range the user must return to CAPWORK and identify the reason the values are so diverse.

As described in the CAPWORK section, the process is now iterative. The performance specialist could make several changes with the model defined as above. He/She might then return to CAPWORK to isolate the OFFICE TEXT. PROC workload into three separate workloads defined as HPDESK, HPWORD, and HPDRAW. KASANDRA could then be invoked a second time to alter the number of active terminals for each application to determine the impact of these changes on the system.

KASANDRA provides several forms of output. A report is output showing utilization, response time, and throughput for each workload. These values can then be graphed in several different formats, the most common being THROUGHPUT versus TERMINALS, RESPONSE TIME versus Terminals, and SERVER UTILIZATION versus TERMINALS. The graphs can be saved as figures to be used in reports explaining the different results produced by the tool.

<sup>2</sup>Simply defined, MPL is the number of active threads of control provided by a workload. For example, a value of 1.2 is used to model an IMAGE data base bottleneck for a single data base application. The value is derived by examining the wake mask of the PCB table. The wake mask reflects the reason a process is waiting. The calculation of MPL assumes that a software constraint can be observed by isolating the cases in which one or more processes are impeded from entering the system due to another processes control of a software resource. The algorithm scans each of the samples from the PCB table looking for points in which one or more processes are in a software wait state. By looking at the number of active processes in the system at these points, and averaging across the number of samples, a maximum limit is established. An MPL of zero indicates no software constraints were found.

	HP 3000	<b>SM04/15</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Performance Measurement for Capacity Planning  
Interex 1987, Vienna, Austria

One question that is often asked by customers is "What if I add ten users of this HP product I haven't purchased yet?". This question is answered by using standard workloads. Standard workloads have been derived by analyzing data from the SUMDB for each HP product. By analyzing all the HPWORD entries in the WORKLOAD data set of the SUMDB, a standard workload can be derived that represents the average HPWORD user. This can then be used by KASANDRA to model the effect of a new application that is added to the system under study.

One resource of the system that KASANDRA cannot model is memory. Memory constraints have two major effects on system performance. First, an upper limit is placed on the number of users on the system. Second, there is overhead induced in the other resources, CPU and disc, to alleviate memory pressure. Memory requirements are difficult to measure because they vary during the life of the workload. A tool is being developed to capture memory requirements of a workload.

### Summary

The SPEP package continues to evolve. New tools are being developed to improve the accuracy of the collection, new reporting tools are easing the burden of data reduction, and new modeling ideas are making the capacity planning efforts more precise. With this package the performance specialist is provided with the tools required to accurately evaluate system performance.

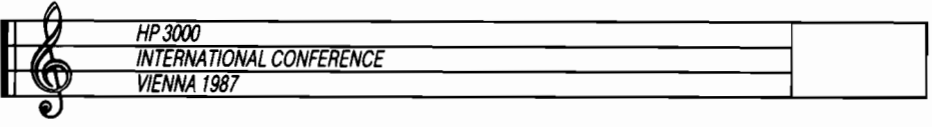
The functions provided by SPEP are being developed for the HP Precision Architecture family of systems. Current collection and modeling tools are being investigated and new tools are being developed to meet the needs of the performance specialists.

### Biography


Tim Twietmeyer is a member of the technical staff for the Performance Technology Center of the Hewlett-Packard Company in Roseville, California. He holds a BS. degree in Computer Science from Chico State University and worked for three years as an HP software engineer and performance specialist in Sacramento. Currently, his efforts are concentrated in working with the SUMDB to define standard workloads for HP products, defining benchmarks for system evaluation, and developing performance tools to support SPEP.

A THANK YOU to the people that are instrumental in the development and support of SPEP, namely Tony Engberg - developer of the original SPEP process and continues to oversee the project team, Paul Primmer - current SPEP project manager and developer of several of the tools used by SPEP, Jim Morris - project manager responsible for field strategy for SPEP, Rick Bowers - currently a project manager, but was the design engineer responsible for the development of KASANDRA, Doug McBride - an engineer responsible for developing SPEP tools and a major contributor to refining the SPEP process, and Gary Hynes - an engineer responsible for the design and development of the CAPLAN data base and CAPWORK. The above mentioned managers and engineers work for the Performance Technology Center which is a part of the Information Technology Group.

A special thanks to the Application Support Division (ASD), especially Marc Hoff and Phil Palmintere. They took SPEP and made it a product so HP customer's can benefit from the package. ASD also continues to support and fund the project team.



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987


	HP 3000	SM05/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## HP Proactive Support Systems Predictive Support and HPTrend

To maintain its industry leading position in support, Hewlett Packard has developed two major products to help customers to maximize the utilization of their systems. Predictive Support and HPTrend, the subjects of this paper, both proactively monitor system performance and automatically report their findings to HP for analysis and action. With these products automatically notifying HP of potential system utilization problems, HP will identify and correct many problems with little or no effort on the part of the system manager. It is with products like these that Hewlett Packard can continue to provide a highly involved and proactive level of support, unequalled in the industry.

This discussion will cover the technical implementation aspects of Predictive Support and HPTrend. The topics covered include the design goals, onsite implementations, data communications, Response Center operations, and next release features of both products.

Presented by Bruce Richards of the HP Knowledge Systems Lab.

	HP 3000	SM05/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	


NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

	HP 3000	SM05/3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## DESIGN GOALS:

Predictive and HPTrend reflect the common idea that HP wants to take action corrective action on potential problems before the customer is ever aware of any difficulty. However, not every problem or bottleneck can be anticipated, therefore, goals were established to measure the effectiveness of the products. Since the volume of data flowing between the customer and HP was going to be significant, the process of handling the data had to be automated on both ends of the transfers. System security for the customer and HP must be maintained. Disc space consumption on the customer system was a very major concern. The products also needed to smoothly integrate with the existing remote support programs.

Predictive Support was given the goal of anticipating 30% of HP 3000 hardware problems. This goal roughly coincides with the occurrence of intermittent failures on electromechanical products, intermittents being the target failure type for Predictive. A subgoal for predictive was to notify HP when it reached a 50-70% certainty that something was going to fail. We did not want 100% certainty since the uncertainty is needed to calibrate our error threshold rulebase. Disc consumption was to be under a megabyte. We wanted to share any error information with the customer, so all messages are displayed for customer review, albeit the messages are somewhat cryptic. Operation of the product, beyond the above, was to be as invisible as possible to the customer. Predictive does not attempt to diagnose the cause of the failures, it only predicts them. Discs, memory, and magnetic tapes were the original products to be supported by Predictive. From early calculations, the cost of developing and implementing Predictive would easily be recovered from field productivity gains alone.

HPTrend's goal was to alert the HP account representative of any potential resource utilization problems on one of his/her sites before the customer had difficulties. A secondary goal was to report the data in an understandable form. Since an HP Systems Engineer was ultimately going to explain the data to most customers, we had to be certain that the data was displayed in a consistent and highly graphic manner. To this end, HPTrend needed to collect onsite data in real-time, periodically send this data to HP, format a consistent and highly graphic report, and distribute it to the HP account team and the customer. Native language support for European users was a must.

Both Predictive and HPTrend were to leverage code as much as possible, especially in the datacomm area. Both needed to minimize human intervention at the HP Response Centers, where the data is received and handled. The products needed to operate similarly (configuration, etc.) to fit smoothly into the customer's daily operations.

## ORGANIZATION OF THE ONSITE SOFTWARE:

### Predictive Support

Products considered for predictive analysis must have a high degree of internal error detection and reporting capability in order for Predictive Support to be effective. Once a candidate has been selected, product experts from HP's Response Centers and manufacturing divisions model the degradation and failure modes of the product. The results are reduced to a set of product independent rules to be incorporated into the Predictive product. The definition of the rules is a dynamic process, so the Response Center experts must pay constant attention to the effectiveness of the rules established for each product.

#### Predictive Support Software Operation

After Predictive is installed on the customer system, the analysis begins. Predictive executes in four basic phases. First, error data is collected. Second, the error data is reduced into a generic format. Third, trend detection is performed, and finally, if necessary, the appropriate actions are taken to solve any problems. Overall processing is controlled by the Predictive monitor process. Fig. 1 shows a graphic representation of the onsite Predictive architecture.

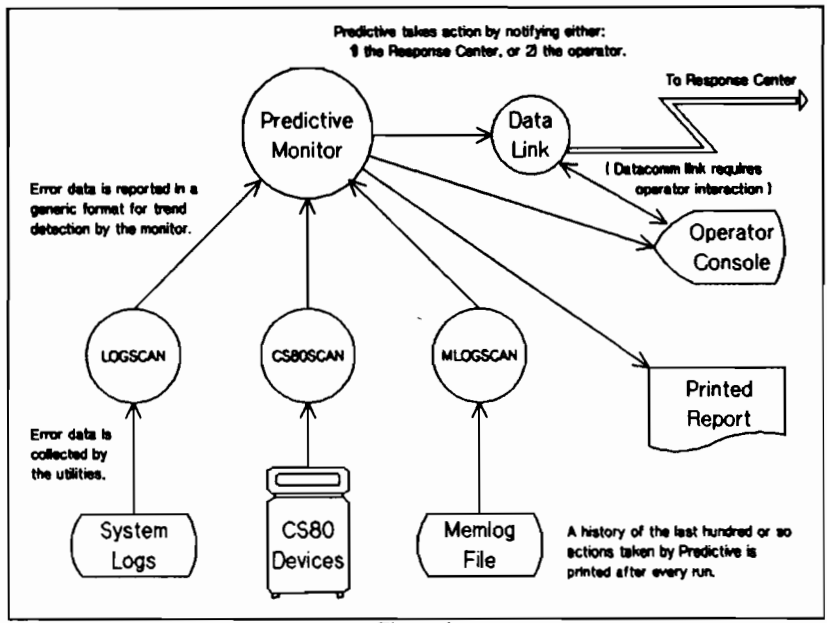



Figure 1

Predictive Support uses special utility programs to collect error data. Each utility is launched as a child process and retrieves the error data for a specific class of products. Predictive Support currently

	HP 3000	<b>SM05/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Onsite Predictive

uses three utility programs to collect data from system-maintained log files, internal disc drive logs, and processor memory logs.

In the second phase of processing, the utilities translate the myriad of error data formats into a common message format, so the Predictive monitor can use the same trend detection algorithm to process the error data to determine whether a failure is imminent on the specific system component. The messages contain information identifying the product and the class of error involved. In addition, error class specific information that is not needed for trend detection, but is necessary for further definition of the particular error, is appended to the message and passed on to the eventual troubleshooter.

When the messages are received by the predictive monitor process for third-phase processing, the error data is passed through a trend detection algorithm. If the results indicate that an undesirable trend has been established, the appropriate action is triggered. For most products, the vast majority of soft errors are normal and transparent to the system users. The trend detection algorithm is described in more detail later.

### Predictive Support Output

Taking the specified action is the fourth and final phase. Predictive notifies the appropriate person so that corrective action can be carried out by the customer or by HP support personnel. In cases where the customer is notified (e.g., a specific medium is marginal, or a drive needs cleaning), the predictive monitor sends a message to the console, informing the operator about the problem. In cases where HP needs to further investigate a potential problem, Predictive Support uses a communication link to transfer the information directly to the Response Center for investigation. At the end of every run of Predictive, regardless of the actions taken, a report is generated listing all of the messages output during this and previous runs of Predictive Support. This provides a hard-copy record of Predictive Support activity.

The data communication link between the customer system and the Response Center uses the remote support modem installed on most of the HP 3000 systems. At the end of the run, if messages have been generated for HP support, Predictive Support requests the system operator's permission to use the modem. Once permission is granted, Predictive autodials the modem, if possible and allowable, calling into a secured port on the Response Center job management system. A single file is then transferred (with error checking) from the customer system to the Response Center. At each step of the transfer the customer has complete control over the connection. More about this data communications software is described later.

At this point Predictive Support will have completed its execution. In the cases where it has transferred action data to the HP Response Center, the HP engineers may investigate the problem further and recommend the appropriate course of action. If this recommendation requires on-site action by the field customer engineer the downtime will be scheduled at a convenient time. This advance warning saves the customer the inconvenience of unscheduled downtime and allows HP to schedule Customer Engineering resources more productively.

### Trend Detection with Predictive

The objective of the Predictive Support software is to monitor trends in computer system operation as they occur, notifying the appropriate person when a trend indicates that there may be a problem. It was obvious that some statistics must be gathered and analyzed, but the questions of how to keep these statistics and what to count had to be resolved.



### Trend Detection Concepts

To analyze system trends, we need to identify entities that can be monitored in hope of detecting signs of impending failures. The most familiar source of failures on a computer system is the hardware, e.g., peripheral devices. The entities that are monitored by Predictive Support are referred to generically as *devices*. Non-peripheral "devices" such as memory are referred to as *pseudodevices*. For each type of device that is to be monitored, some set of significant observable events must be identified, so their occurrence can be tabulated and analyzed. The best example of an event to be monitored is an I/O error condition, such as a recovered read error on a disc drive. As implied by the designation, these errors are detected and corrected by the drive and do not have a noticeable effect on performance or integrity. A single occurrence of this event would not indicate a problem, but when the frequency of recovered error occurrence reaches a certain level, a pending failure may be indicated.


For trend detection, the frequency of occurrence of these significant events must be calculated for each device being monitored. Merely tabulating the occurrences of an event will not, however, yield the frequency of occurrence. We therefore need to track some other factor to weight the occurrence of the significant event. Since simple continuous tracking of event occurrence and weighting factor will lead to a dilution of the statistics, the accumulation of the weighting factor must be limited to create a sample across which the frequency of occurrence can be analyzed. The sample can count something related to the event, such as the number of disc accesses, or a time interval. To identify an undesirable trend, a threshold must be defined where the occurrence value, relative to the defined sample size, is considered undesirable.

In summary, to detect abnormal system trends, we need: 1) entities to monitor, that is, devices and pseudodevices, 2) definition of significant events that occur on the devices, and 3) rules defining both the samples across which event frequencies are monitored and the relative thresholds of event occurrence where problems may be indicated.

### Trend Detection Data Needs

The Predictive Support software uses two major data structures to manage the collection and analysis of the statistics kept for detecting system trends. One is the trend log, a repository for collected event occurrence data. The other is the normalization matrix which contains a set of rules governing the logging of event data in the trend log and the interpretation of the resulting trend data. The matrix makes the trend detection process table-driven. There is a unique set of events defined for each type of device monitored and one or more rules in the matrix for each event. A single element of the trend log is referred to as a bucket and contains the current event occurrence and sample values, along with a time stamp associated with the last time the bucket was updated. There is logically one bucket in the trend log for each normalization rule associated with each actual device or pseudodevice configured on the system. The buckets are grouped into sets by device type. The set size is determined by the number of devices configured as the given type, leading to a one-to-one correspondence between rules and bucket sets.

The rule has two parts that control the collection and analysis of statistics in the buckets of its set: a bucket definition and an action specification. There are fields in the bucket definition that define the bucket's sample value in terms of the normalization method and maximum size. The method specifies whether the sample is a time interval or an accumulation of some other event (e.g., disc accesses). The sample value in the bucket cannot exceed the maximum size defined in the rule. If the logging of event data causes this condition to occur, the occurrences relative to the obsolete portion of the sample must be discarded before the new event occurrences can be added in. The formula used to reduce the occurrence value when this overflow condition arises is described in detail on page 8. The bucket definition also includes the event occurrence threshold. If after event data is logged the

	HP 3000	<b>SM05/7</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Onsite Predictive

bucket's event occurrence value is at or above the threshold, the action defined by the action specification part of the rule will be taken.

The actions of the Predictive Support system will frequently involve communication with human users, namely the customers and/or HP's support engineers. After the message is sent to the appropriate person, there are three options: the bucket values (time stamp omitted) can be left alone, reset to zero, or reset with event processing suppressed (only if the bucket has a time normalization method). For example, some disc devices have their own microprocessor controller for which the software is held in ROM. The utility for examining their logs only supports the current (and future) ROM revision, and it checks the revision level before reading the drive logs, reporting an event if the customer's drive is using an unsupported version. This condition is then reported to the Response Center, and a service call is initiated to upgrade the ROM. Service calls of this nature are usually scheduled weeks in advance, so there is no need to continue reporting it each time Predictive Support is run. For this reason, event processing is suppressed for that bucket for the length of the sample size (usually 3 weeks).

### Predictive Event Processing

During event processing, the predictive monitor launches each utility, one at a time, and waits for it to send a message. It wakes up when a message is received, and if it is an event record, the following algorithm is performed to log the trend data:

Use the device identification information to find the first rule in the normalization matrix.

Read the trend log values associated with the rule, and use the time stamp, along with the event record's time stamp, to compute the interval to be used as the event record's sample value for rules with a time normalization method.

FOR EACH rule associated with the event number and device type specified in the event record, log the event record's trend data into the trend log as follows:

IF the rule has a suppress action,  
 AND the bucket is suppressed,  
 AND the time since last update is less than the rule's maximum sample,  
 THEN skip the following steps, continuing with the next rule.

Use the algorithm described in the next section to log the event data into the trend log.

IF the bucket's new event occurrence value is greater than or equal to the threshold defined in the rule,  
 THEN take the specified action by outputting the message to the appropriate destination and optionally resetting the bucket values and suppressing further event processing.

Continue with the next rule and, when all rules have been applied, wait for the next message.

## Logging Event Data in the Trend Log

Logging event data involves adding the event record values to the current trend log values. The resulting event occurrence value is then compared with the threshold to determine whether to take action.

Three pairs of values are used for trend detection. X values are associated with event occurrences and Y values are associated with the sample values. In this description, each of the pairs will have a subscript designation. The count and weight values from the event record use the subscript e. For example, if the normalization method is time,  $Y_e$  = time since last update. The trend log values for current event occurrence and sample values stored in a bucket use the subscript b. The threshold and maximum sample values defined in the normalization rule use the subscript r.

When an event record's values are logged, one of three mutually exclusive conditions exist. The equations used to set the new bucket values in each case are as follows:

CASE1:  $Y_e > Y_b$  (event sample larger than maximum)

$$\begin{aligned} X_b &<- (Y_r)X_e/Y_e \\ Y_b &<- Y_r \end{aligned}$$

CASE2:  $(Y_e + Y_b) \leq Y_r$  (bucket sample will not overflow)

$$\begin{aligned} X_b &<- X_b + X_e \\ Y_b &<- Y_b + Y_e \end{aligned}$$

CASE3:  $(Y_e + Y_b) > Y_r$  (bucket sample will overflow)

$$\begin{aligned} X_b &<- X_b - (Y_b + Y_e - Y_r)(X_b/Y_b) + X_e \\ Y_b &<- Y_r \end{aligned}$$

Under Case 1 the event record values cover the entire sample, so the old statistic is discarded and the ratio of the event record values is multiplied by the maximum sample, resulting in the new  $X_b$  value, an approximation of the number of occurrences over the required sample.

Under Case 3 an attempt is made to discard obsolete occurrences, so the statistic will only be evaluated for the sample specified in the rule. The  $X_b$  value must be normalized before  $X_e$  is added in. The amount of overflow is multiplied by the current ratio of the bucket (i.e., frequency of occurrence), and the result, an approximation of the number of obsolete occurrences, is subtracted from  $X_b$  before  $X_e$  is added in.

## ORGANIZATION OF THE ONSITE SOFTWARE: HPTrend

### Function of HPTREND

HPTREND software performs data collection on the customers HP 3000 computer system. The data saved by HPTREND is then transported to a Response Center for processing. There it is transformed into a series of charts and graphs which describe how the customer is using the HP 3000. The reports generated provide the customer with information that will enable him/her to perform resource planning or application sizing, thereby making both the HP 3000 and the customer's users more productive. Figure 2 describes at a high level, how HPTREND works overall.

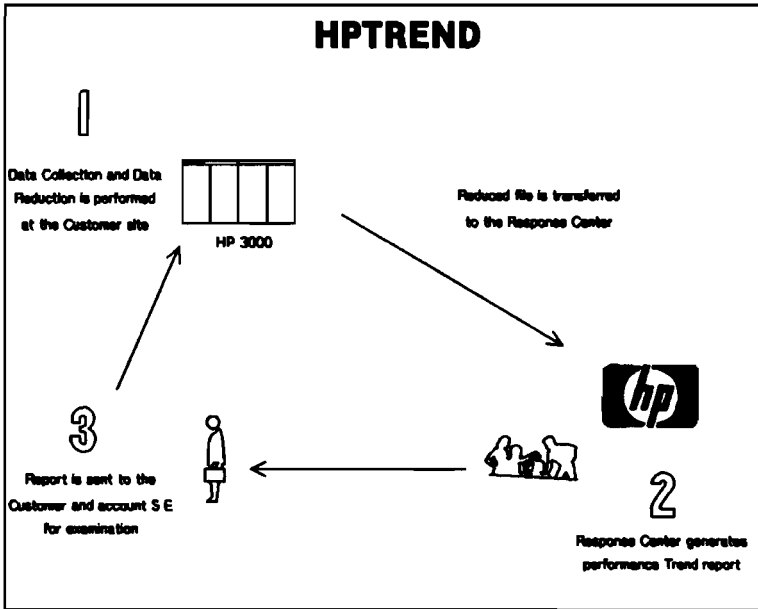



Figure 2

As shown in figure 2, HPTREND runs on the customer system performing data collection (from the system logfiles and the Measurement Interface), data reduction and data transfer to an HP Response Center. The Response Center receives the HPTREND data and produces reports. Finally, the HPTREND performance report is sent to the customer (or delivered by the Account Representative).

	HP 3000	<b>SM05/10</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Onsite HPTrend

## HPTREND's Product Environment

HPTREND can be installed on any HP 3000 computer system which is running an MPE release other than V/P or V/P-Delta-1. HPTrend lives in the TELESUP account in it's own group (HP35136A).

To be effective in it's task, HPTREND requires that system logging be enabled and that certain MPE log file records be logged (in particular, 1,2,3,5,6 and 8). Note that if the logging is not turned on, HPTREND will give incomplete reports and as a result be quite ineffective in it's task.

HPTREND also requires a 1200 BAUD modem to use in the transfer of HPTREND data to the Response Center for processing. This modem can be autodial or non-autodial and is shared with Predictive Support and Remote Support uses.

## Overview

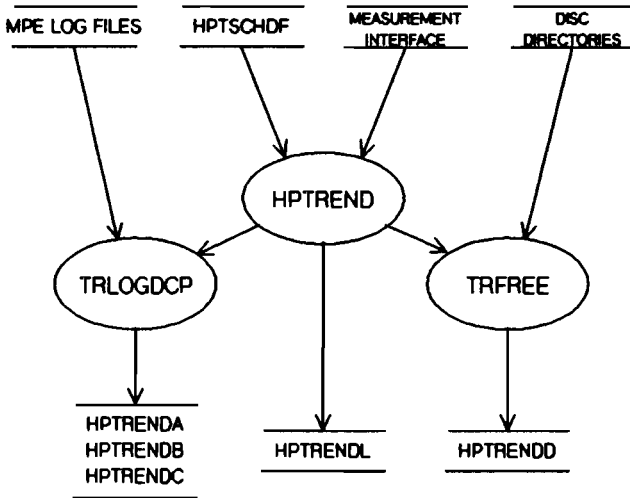
HPTREND is a product which may be used to produce a performance trend report on the HP 3000. It consists of two phases:

- o Data collection and data reduction phase.
- o Report generation phase.

The data collection and data reduction is performed at the customer site. HPTREND software runs at all times, gathering data from the MPE Measurement Interface and MPE Log files. The Measurement Interface data collection is performed by the program HPTREND. This program launches the MPE Log file program TRLOGDCP at a specific time (the default being midnight) to collect Log file data. This program also launches the Disc Free Space capture program 'TRFREE' each time the log file data capture program 'TRLOGDCP' is run. Once a day the HPTREND program looks at the schedule file to see if it's time for Data reduction. The schedule file (HPTSCHDF) is a list of dates when data is to be reduced and sent to the Response Center. When the first date in the schedule file is equal to the current date, the HPTREND program launches the Data reduction Job stream JREDUCE. This job stream runs the reduction program REDUCE. The JREDUCE job stream then runs HPTXFDMP to transfer the reduced file to the Response Center. Once the file is transferred to the Response Center. The second phase of HPTREND (report generation) produces the performance trend report and then this report is sent to the customer and HP account rep for examination.



Onsite HPTrend

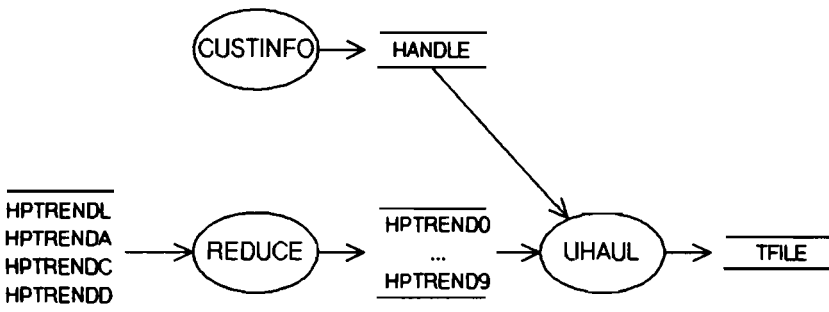


**NOTE**

The HPTREND program can create and run the TRLOGDCP and TRFREE programs as son processes automatically at 24 hour intervals. This allows all data collection to be done via a single batch job. The programs are inherently interruptable and restartable to simplify data collection operations.

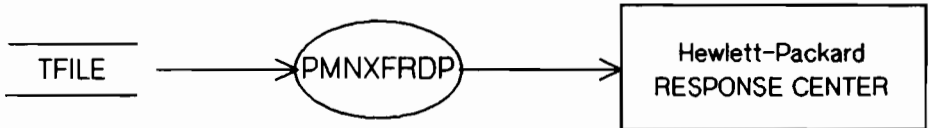
**DATA REDUCTION:**

Data reduction need only be done once for each report generated. It takes less than two minutes on the typical customer system.



## DATA TRANSFER:

Data transfer program PMNXFRDP takes the TFILE and transfers this file to the Response Center.



## Program Descriptions

The following programs are used on the customer site. Functional descriptions of the program files are provided along with a description of any data files used. All program files reside in the HP35136A group of the TELESUP account on the customer's system.

### HPTREND.

This program runs on the customer's system and is part of the data collection phase. It enables the measurement interface (global statistics only) and logs data to the HPTRENDL file as follows:

Every five minutes HPTREND awakes after calling the PAUSE intrinsic. At this time the global statistics are captured and differences are calculated from their values before pausing. TIMER is called to get elapsed time and eliminate problems with inaccurate redispaching. The individual counts from the measurement interface are turned into rates per unit time and converted to normalized units. Certain items are combined into general categories. Next, running averages of these categories and maximum rates are accumulated each ten minutes until the hour of the day changes. At this time a summary record is appended to the HPTRENDL disc file and the internal accumulators are reset. Each record in the disc file is date and time stamped for later analysis. The records are also encrypted before being written so as to make pirate report generation more difficult. Special procedures are used to handle counter roll over and the effects of multiple users of the measurement interface. Duplicate HPTREND programs are allowed but only one can write to a particular HPTRENDL file at a time.

At a specified time (the default being midnight) HPTREND will schedule the TRLOGDCP program to run as a son process. It runs TRLOGDCP without wait and will not abort if it is not present or runnable. The user may specify when TRLOGDCP is run through the use of the ;PARAM= parameter on running HPTREND. Simply specify the desired hour to run in 24 hour time (0-23). Any parameter outside the range (0-23) will cause HPTREND to NOT RUN TRLOGDCP. In this case TRLOGDCP should be run manually. HPTREND will run the Disc Free Space Capture program 'TRFREE' each time the 'TRLOGDCP' log file data capture program is run.

Once a day relative to when the HPTREND job stream was launched, HPTREND will look at the schedule file. The schedule file (HPTSCHDF) is list of dates and times when data is to be reduced and sent to the Response Center. If the first date in the Schedule file is equal to the current date, and the time is equal to current time, the data reduction job stream (JREDUCE) is streamed programmatically. Several error conditions can occur.

#### TRLOGDCP.

This is the data collection program for the MPE LOG file data. It can be run as a son process by HPTREND or run manually on demand. It asks no questions and so can run entirely without operator intervention. The only time an operator will need to interact with TRLOGDCP is if he or she asks it to read log files from Magnetic Tape.

TRLOGDCP gathers information from the MPE Log Files using the following log record types:

- 2 -- Job Initiation.
- 3 -- Job Termination.
- 5 -- File Closes.
- 6 -- System Shut Down.
- 8 -- Spool File Done.

In order to obtain useful information the user must enable logging for at least these events on the customer's system. The individual log records are summarized in two ways: day by day activity (CPU, DISC, PRINT, TAPE, JOBS) is written to file HPTREND and the name of the ACCOUNT that used the resource (same activities as above plus CONNECT time) is written to file HPTRENDA. File HPTRENDB is a scratch file used by TRLOGDCP to pick up analysis when it is restarted.

The normal operation for TRLOGDCP is as follows:

Whenever it is run, TRLOGDCP will determine if there are any MPE Log Files in PUB.SYS that it has not yet analyzed. If so it will summarize them and update its data files to indicate the first and last log file analyzed. In this mode TRLOGDCP requires no operator intervention and analyzes only disc log files. The program uses Privileged Mode in order to allow it to read log files in spite of MPE Security. It does not alter the files nor the security on them. Data kept in the HPTREND files is encrypted.


A "system ID" can be supplied using the ;INFO= string in the RUN command. This system ID is passed to the reporting system where it can be included in the report. (If TRLOGDCP is run by HPTREND then HPTREND passes it's ;INFO= string along to TRLOGDCP).

This program analyzes approximately 140,000 log file records per minute on an unloaded system. As such it should normally not run for more than a few minutes unless it has several months of log files to analyze.

#### REDUCE.

This program takes the data files created by HPTREND (HPTRENDL), by TRLOGDCP (HPTRENDA and HPTREND), by TRFREE (HPTREND) and further summarizes them into only that data necessary to produce the HPTREND report. REDUCE should be run just prior to transmitting the data files for report generation. A Job stream (JREDUCE) has been provided to simplify data reduction and compaction.



	HP 3000	<b>SM05/14</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Onsite HPTrend

The REDUCE program creates ten small data files (HPTREND0 through HPTREND9) which, once decrypted, are used directly by the HPTRPLOT program to produce the report plots. Default operation is simple, just RUN REDUCE. If the output files already exist, they will be purged and rebuilt. Data reduction time is normally less than two minutes.

The REDUCE program has special entry points which are used to determine how much data should be included in the final report. Normally the hourly HPTREND data and daily TRLOGDCP data are not purged by REDUCE but rather left to accumulate. This is to allow longer interval reporting at a later date if desired. If the REDUCE program is run with the "RESET" entry point

```
:RUN REDUCE,RESET
```

it records a 'reset date' in the data files. Subsequent runs of REDUCE will only summarize data on or after this date. By default the reset date is the date at the time REDUCE is run. You may specify another date through the use of the ;PARM= parameter. The PARM= value is the number of days OFFSET from today which you want to use as a reset date. For example:

```
:RUN REDUCE,RESET;PARM=-30
```

would set the reset date to thirty days before today's date. Note that when run with the RESET entry point, no actual data reduction is done.

You may select dates to be analyzed that differ from the RESET date by running REDUCE;PARM=-nnn. This feature may be used to produce year end plots covering more than the standard dates. Note that this option does not change the RESET date, it merely overrides it temporarily.

**IMPORTANT NOTE** : The MPE Log file data pertaining to ACCOUNT usage is a cumulative file with no internal date coding. (This was done in order to drastically reduce the amount of disc space required). Whenever REDUCE is run with the RESET option, this file is PURGED and REBUILT. The next time TRLOGDCP is run it will begin adding account information with the first log file it analyzes. The actual dates covered in this file (HPTRENDA) are recorded in it and reported on the final report. Be aware that this part of the report data may not always exactly match the other data files but the report will indicate the dates covered in each category.

You may list the reset dates as well as other data collection information by running REDUCE with the LIST option. This option does NOT reduce data.

```
:RUN REDUCE,LIST
```

**UHAUL.**

The Uhaul program combines the various datafiles, combines them into one file, and compresses the file for transmission. A similar program on the Response Center side will decompress and expand the files into their original structure. The TFILE is this combined and compressed file.

## HPTrend File Contents:

Note: The description of the data files are included to give the reader an idea of what data is collected and to aid in his/her understanding of HPTrends measurements. The actual onsite files are encrypted to maintain data integrity.

## HPTRENDL

HPTRENDL contains the detailed Measurement Interface Data. One record per hour when HPTREND was running 20 word records containing binary data.

Type	Contents
L	DATE (CALENDAR format)
D	TIME (CLOCK format)
I	Average CPU busy TOTAL (percent times 10)
I	Maximum CPU busy TOTAL ( " )
I	Average Paused for Disc ( " )
I	Maximum PAUSED for Disc ( " )
I	Average BUSY on USER ( " )
I	Maximum BUSY on USER ( " )
I	Average MAM CPU ( " )
I	Maximum MAM CPU ( " )
I	Average CACHE CPU ( " )
I	Maximum CACHE CPU ( " )
I	Average LOGICAL DISC IO (rate/second times 10)
I	Maximum LOGICAL DISC IO ( " )
I	Average PHYSICAL DISC IO ( " )
I	Maximum PHYSICAL DISC IO ( " )
I	Average MAM DISC IO ( " )
I	Maximum MAM DISC IO ( " )
I	Number of data samples for this hour
I	Average Terminal read rate per minute (times 10)
I	Maximum Terminal read rate over the hour

Data is taken at five minute intervals. The AVERAGE and MAXIMUM's are calculated for each hour based on these five minute samples.

**CPU busy TOTAL** = 100% - IDLE time - Background Garbage Collection  
**PAUSED for DISC** = Paused for SWAP + Paused for DISC + Paused for BOTH  
+ Paused for CACHE  
**BUSY USER** = BUSY on USER processes  
**MAM CPU** = MAM + Local garbage Collection  
**CACHE CPU** = CACHE on user stack + CACHE on ICS  
**LOGICAL DISC IO** = GLOBAL READS + GLOBAL WRITES  
+ (for each disc) READS (blocked/unblocked/iowait)  
+ WRITES (blocked/unblocked/iowait)  
**PHYSICAL DISC IO** = (for each disc) READS (blocked/unblocked/iowait)  
+ WRITES (blocked/unblocked/iowait)  
+ CACHE READS  
+ CACHE background writes + CACHE forced writes  
**MAM DISC IO** = (for each disc) code reads + data seg reads  
+ data seg writes + forced data seg writes

Counts are converted to percentages or rates by dividing by the elapsed time for each sample. Note that all numbers are inflated by a factor of ten to increase precision while still using only a 16 bit integer to store the number. You must divide the raw numbers by ten to get the actual rate or percentages.

**HPTREND0**

(Disc Free Space data, created from HPTREND0 by running REDUCE) This file contains 32 word records of binary data.

Type	Contents
L	DATE (CALENDAR format)
D	TIME (CLOCK format)
I	Number of Discs on the system
R	Largest Free Disc Space (Ksectors)
R	Total Disc Free Space (Ksectors)
D	Count of holes 1-9 sectors
R	Disc space in holes 1-9 sectors (Ksectors)
D	Count of holes 10-99 sectors
R	Disc space of holes 10-99 sectors (Ksectors)
D	Count of holes 100-999 sectors
R	Disc space of holes 100-999 sectors (Ksectors)
D	Count of holes 1000-9999 sectors
R	Disc space of holes 1000-9999 sectors (Ksectors)
D	Count of holes 10,000-99,999 sectors
R	Disc space of holes 10,000-99,999 sectors (Ksectors)
D	Count of holes >= 100,000 sectors
R	Disc space of holes >= 100,000 sectors (Ksectors)

**HPTREND1**

(Reduced Measurement Interface data, created from HPTRENDL by running REDUCE) This file contains 26 word records of binary data, one record per day.

Type	Contents		
L	DATE (CALENDAR format)		
I	TOTAL CPU	ALL SHIFTS	(percent times 10)
I	"	1st Shift	(" )
I	"	2nd Shift	(" )
I	"	3rd Shift	(" )
I	"	4th Shift	(" )
I	USER CPU	ALL SHIFTS	(" )
I	"	1st Shift	(" )
I	"	2nd Shift	(" )
I	"	3rd Shift	(" )
I	"	4th Shift	(" )
I	LOGICAL DISC	ALL SHIFTS	(rate/sec times 10)
I	"	1st Shift	(" )
I	"	2nd Shift	(" )
I	"	3rd Shift	(" )
I	"	4th Shift	(" )
I	PHYSICAL DISC	ALL SHIFTS	(" )
I	"	1st Shift	(" )
I	"	2nd Shift	(" )
I	"	3rd Shift	(" )
I	"	4th Shift	(" )
I	MAM DISC	ALL SHIFTS	(" )
I	"	1st Shift	(" )
I	"	2nd Shift	(" )
I	"	3rd Shift	(" )
I	"	4th Shift	(" )
I	PAUSED FOR DISC	ALL SHIFTS	(rate/min times 10)
I	"	1st Shift	(" )
I	"	2nd Shift	(" )
I	"	3rd Shift	(" )
I	"	4th Shift	(" )
I	TERMINAL READ RATE	ALL SHIFTS	(" )
I	"	1st Shift	(" )
I	"	2nd Shift	(" )
I	"	3rd Shift	(" )
I	"	4th Shift	(" )

Where "ALL SHIFTS" means all day  
 "1st Shift" is Midnight - 6:00 AM  
 "2nd Shift" is 6:00 AM - Noon  
 "3rd Shift" is Noon - 6:00 PM  
 "4th Shift" is 6:00 PM - Midnight

Each category is an average of the AVERAGE values from HPTRENDL

### HPTREND2

(Reduced Measurement Interface data, created from HPTRENDL by running REDUCE.)

This file contains exactly 25 records of 11 words each and is BINARY data. One record is for each HOUR of the day (0-24) with the first record duplicated into the 25th (so the plots form a circular plot).

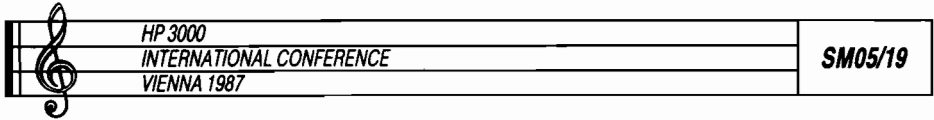
Type	Contents
I	HOUR of the day (0-24) << the following numbers are for weekdays only MON-FRI >>
I	TOTAL CPU used (Percent times 10)
I	USER CPU ( " )
I	LOGICAL disc IOs (Rate/sec times 10)
I	PHYSICAL disc IOs ( " )
I	MAM disc IOs ( " )
	<< the following numbers are for weekends only SAT,SUN >>
I	TOTAL CPU used (Percent times 10)
I	USER CPU ( " )
I	LOGICAL disc IOs (Rate/sec times 10)
I	PHYSICAL disc IOs ( " )
I	MAM disc IOs ( " )
I	Paused for Disc ( rate/min times 10) << Weekdays >>
I	Paused for Disc ( " ) << Weekends >>
I	Terminal Read Rate ( " ) << Weekdays >>
I	Terminal Read Rate ( " ) << Weekends >>

Again, these numbers are averages of the HPTRENDL averages only this time they are accumulated by the hour of the day rather than by the day of the year (Each record contains the average usage for that time interval over all days in the HPTRENDL file).

### HPTREND3

This file is produced by the REDUCE program from HPTRENDL. It contains one record per day and is in BINARY format.

Type	Contents
L	DATE (CALENDAR format)
I	# of JOBS & SESSIONS
D	# of DISC IOs
D	# of pages printed
D	# of tape IOs
D	# of CPU Seconds at BS priority
D	# of CPU Seconds at CS priority
D	# of CPU Seconds at DS priority
D	# of CPU Seconds at ES priority
D	# of CPU Seconds by Interactive Sessions
D	# of CPU Seconds by Batch Jobs



Onsite HPTrend

#### HPTREND4

This file is produced from HPTRENDA by the REDUCE program. It contains 11 records, one for each of the top ten accounts plus one for all other accounts (account name = (OTHERS).) It is sorted by CPU usage.

Type	Contents
char	ACCOUNT
R	Total CPU hours
R	CPU hours by Interactive Sessions
R	CPU hours by batch jobs

#### HPTRENDS

This file is produced from HPTRENDA by the REDUCE program. It contains 11 records, one for each of the top ten accounts plus one for all other accounts (account name = (OTHERS).) It is sorted by CONNECT time.

Type	Contents
char	ACCOUNT
R	Connect Hours (Batch + Interactive)

#### HPTREND6

This file is produced from HPTRENDA by the REDUCE program. It contains 11 records, one for each of the top ten accounts plus one for all other accounts (account name = (OTHERS).) It is sorted by DISC IOs.

Type	Contents
char	ACCOUNT
R	# of DISC IOs

#### HPTREND7

This file is produced from HPTRENDA by the REDUCE program. It contains 11 records, one for each of the top ten accounts plus one for all other accounts (account name = (OTHERS).) It is sorted by Mag Tape IOs.

Type	Contents
char	ACCOUNT
R	# of mag tape IOs



Onsite HPTrend

### HPTRENDS

This file is produced from HPTRENDA by the REDUCE program. It contains 11 records, one for each of the top ten accounts plus one for all other accounts (account name = (OTHERS).) It is sorted by Pages Printed.

Type	Contents
char	ACCOUNT
R	# of pages printed

### HPTREND9

This file is produced from HPTRENDA by the REDUCE program. It contains 11 records, one for each of the top ten accounts plus one for all other accounts (account name = (OTHERS).) It is sorted by total number of Jobs & Sessions.

Type	Contents
char	ACCOUNT
D	# of sessions
D	# of jobs

### HPTRENDA

This file is produced by the TRLOGDCP program. It contains one record for each ACCOUNT found on the system. Each record is the accumulation of activity on the account since the last time REDUCE was run (unless the NORESET entry point was used). REDUCE uses this file to produce files HPTREND4-HPTREND9.

Type	Contents
char	ACCOUNT
D	# of sessions
D	# of jobs
R	# of disc IOs
R	# of pages printed
R	# of tape IOs
R	connect hours (batch & interactive)
R	CPU Hours for Interactive sessions
R	CPU Hours for Batch jobs

### HPTREND8

This file is temporary storage for the TRLOGDCP program. It contains a record for each active job or session at the termination of the last running of this program. It's main purpose is to correlate the job/session numbers to the accounts under which they logged on.

Type	Contents
L	JOB/SESSION number (MPE LOG FILE format)
L	Index into local storage (not used in the file)
char	Execution priority ("B", "C", "D", "E" in the left byte)
L	Job State flag
	12:1 -- Job Initiated
	13:1 -- Job Terminated
	14:1 -- STDLIST closed
	15:1 -- STDIN closed
char	ACCOUNT

### HPTREND3

This file is produced by the TRLOGDCP program and contains day by day data. Its format is exactly the same as the HPTREND3 file.



**HPTREND0**

This file is produced by the TRFREE program and contains Disc free space data. REDUCE uses this file to produce file HPTREND0. This file is built circularly with room for approximately 2 years of data.

Type	Contents
L	DATE (CALENDAR format)
D	TIME (CLOCK format)
I	Number of Discs on the system
D	Largest free disc space (sectors)
D	Total disc free space (sectors)
D	Count of holes 1-9 sectors
D	Disc space in holes 1-9 sectors
D	Count of holes 10-99 sectors
D	Disc space in holes 10-99 sectors
D	Count of holes 100-999 sectors
D	Disc space in holes 100-999 sectors
D	Count of holes 1000-9999 sectors
D	Disc space in holes 1000-9999 sectors
D	Count of holes 10,000-99,999 sectors
D	Disc space in holes 10,000-99,999 sectors
D	Count of holes >= 100,000 sectors
D	Disc space in holes >= 100,000 sectors

## PROACTIVE DATACOMM SOFTWARE:

To make these proactive products a success, the ability to automatically send data from the customer site to the HP Response Center was a must. The HP 3000 had no low cost (free) datacomm software that operated in an HP 3000 to HP 3000 mode, necessary for both products. There was also the need for a generic file transfer vehicle to be built to download patches and updates to the customer system. As a result, an error corrected, autodialing, file transfer program was written using DSNLINK 100 as a base. It is a generic utility capable of copying any type of file between any two HP 3000's where the software is installed.

### PMNXFRDP.

PMNXFRDP is a program developed to make use of the autodial capabilities of certain modems in use by Hewlett-Packard, and to allow a cost effective method of transferring a file from one HP 3000 to another HP 3000. In the case of HPTREND, this capability allows HPTREND report generation data to be transferred in a timely fashion to a Response Center for report production. Predictive Support transfers event data to the Response Center Predictive Support history database and job management systems.

There are a number of requirements associated with the use of this program. The foremost requirement is that the program must exist on both the controller and host systems. This is not unusual; most inter-system file transfer programs have supporting code on both systems (ex. DSN/LINK100 for HP PC's). The other requirements relate to the use of the outward auto-dialer and the modem used for the interconnection.

The PMNXFRDP program is designed to be initiated in a number of ways. It has the capability to be run in either job or session mode, and can be initiated using the "CREATEPROCESS" MPE Intrinsic. For use with HPTREND, PMNXFRDP is initiated by using the "CREATEPROCESS" MPE Intrinsic through the retry monitor (HPTXFDMP) in job mode.

The PMNXFRDP program has been designed to handle problems in either the transfer or dialing portion of the problem. If for some reason problems are encountered in either portion, an error code will be returned and an abnormal (but expected) termination will result. The error information is returned to the \$STDLIST of the job stream from which PMNXFRDP is run.

The PMNXFRDP program interfaces to many aspects of the system and maintains a good deal of communication with the user. The program code is actually broken up into 3 major components. These are:

1. **Control**, which is responsible for most of the user and system operator dialogue as well as error reporting;
2. **Dial**, which will perform the auto-modem sensing, dialing the remote computer, logging on to the remote system, and initiating the transfer program; and
3. **Xfer**, into which all the actual file transfer protocols and capabilities are built.

The way in which the PMNXFRDP program works can best be explained by illustration. Figure 3 shows how a typical file transfer between two HP 3000's takes place.

Proactive Data Communications

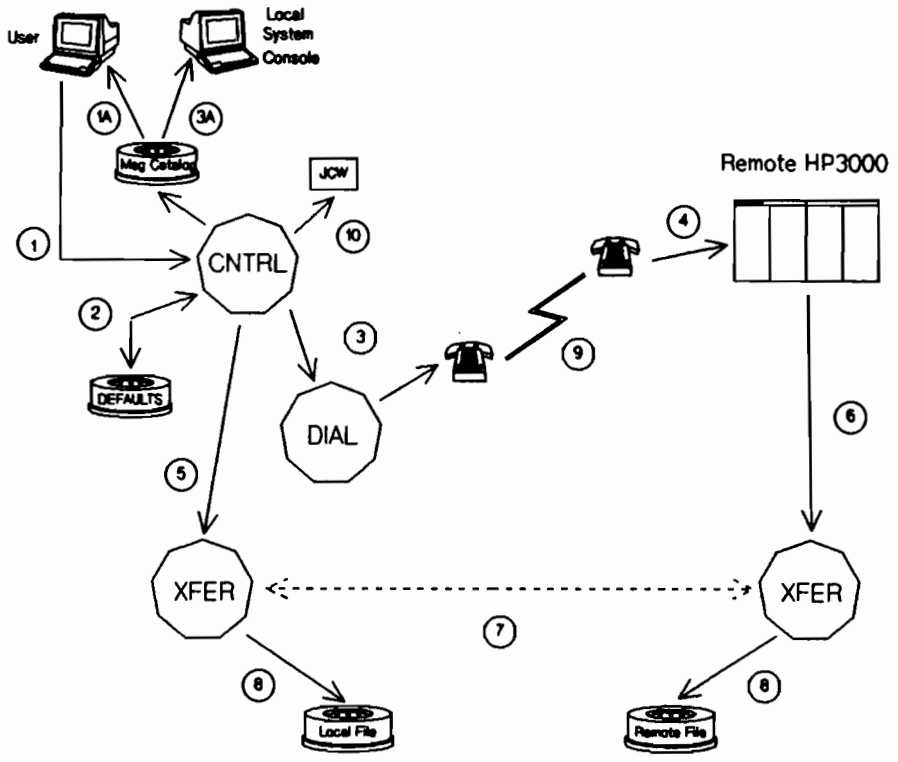


Figure 3

The following information applies to figure 3 and explains the steps in making a data transfer:

- (1) The PMNXFRDP program is initiated through the MPE RUN command or its equivalent. (HPTREND uses the CREATEPROCESS via the Retry Monitor, HPTXFDMP.)
- (1A) Using a local-language message file (PUTXFRDM), we will obtain the values we need to run this program from the job stream. The dialogue begun here will continue throughout the program.
- (2) A file containing default values for running the program may optionally be accessed. A provision to save the values currently being used is also available (NOT USED UNDER HPTREND).
- (3) Using the DIAL procedure, auto-sense the modem and initiate a phone call to the remote system.
- (3A) If the modem is non-auto-dialing, it may be necessary to involve the system operator. In this case, we will communicate to him/her through the message file.
- (4) Once the system-to-Response Center link is established, the customer is validated through the use normal logons (psudo logons for HPTrend and Predictive Support).
- (5) On the customer's system, pass control to the XFER portion of the program.
- (6) At the Response Center, prepare to receive the data file.
- (7) The link is established and both systems are ready to transfer the data file. Commence with the file transfer.
- (8) On the customer system, the data will be read from the FROMFILE and transferred to the Response Center.
- (9) The file transfer is now complete. The customer system informs the Response Center Software and the telecommunications link is brought down.
- (10) Inform the user of the success or failure of the transfer operation through a local-language message and also through a JCW (Job Control Word) which is used to inform the user.

The local-language message file is used to communicate with both the user and the console operator. It is created and accessed via the MPE message system as specified in the MPE Intrinsic Manual.

As stated earlier, this program is best understood if broken into three component parts. These are expanded on below.

### Control Procedure

Upon initial entry into the program, the message catalog is opened and the information concerning what is to be transferred and where is determined (where the customer's system is always the Controller); transfer control information must be received from the job stream. This is done in the control procedure of PMNXFRDP.

When PMNXFRDP is first launched, it displays its startup banner to \$STDLIST (the JREDUCE job listing) as follows:

```

PMNXFRDP (A.02.02) MPE V - SUPPORTED CONTRIBUTED UTILITY
Copyright (c) Hewlett-Packard Co.1985,1986 All Rights Reserved.
TUE, NOV 25, 1986, 3:06 PM

```

This program transfers files between systems using  
asynchronous modems/hardwired communications.

If the message catalog cannot be opened, the following message will be displayed in the job listing (and possibly to the customer's console):

Missing Msg: Set = ## Num = ####

If this occurs, the message file (PUTXFRDM) is either not valid as a message catalog, or it has been purged. PUTXFRDM must be in place for HPTREND to work correctly.

PMNXFRDP will now attempt to get the required transfer information from the JREDUCE job stream. The output will appear as follows if the information is correct:

```
Do you wish to use the "Defaults" File (N/Y)?
You are running this program from the Controller.
Will the Host or Controller receive the data transfer (Host)?
"FROM" filename?
"TO" filename (TFILE.HP35136A)?
Host system phone number?
Logon code for host system?
Port number of out-dialing modem?
```

If all the information is correct, the dial procedure will be called.

### Dial Procedure

The Dial procedure is responsible for the following:

- o Modem port preparation and verification.
- o Modem interrogation.
- o Logging on to the Response Center system.

Dial will first open the Logical Device to which a 1200 BAUD Bell 212 type modem is supposedly attached. The following modems are supported for autodial by PMNXFRDP:

- o Support link 1

- o Support link 2
- o South Queensferry 37212A

Modem attention characters are pumped to the Modem port (with respect to the above order of modems) until a proper response is sent back from a modem. If a supported modem cannot be found, the modem is assumed to be non-autodial. Special requirements are placed upon PMNXFRDP in the way terminal devices (like modems) can be configured on the HP 3000 Computer system. These requirements are described in the HPTrend or Predictive User's Guides.

PMNXFRDP is equipped with a separate autodial modem driver for each different modem type. Once a modem has been identified, the corresponding modem driver is used to perform the actual dial operation. One of the following messages will print on the job listing if the dial is successful:

```

**** Dialing on Support Link ****
**** Dialing on Support Link II ****
**** Dialing on 37212A ****
**** Please respond to the console request ****

```

In the case of a non-autodial modem being used, the following request will be sent to the customer's system console requesting someone to dial the modem:

```

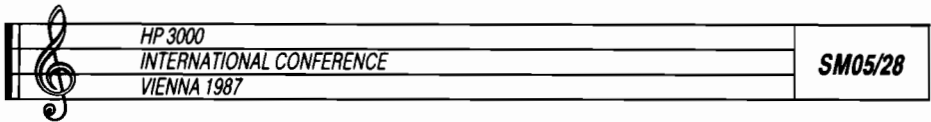
*
* In order to use the assigned modem, the
* operator is needed to dial the telephone.
*
* Please check that the modem on Ldev ## is
* is in HIGH SPEED mode, dial #####,
* place the modem ON LINE and Reply Y to
* the console request.
*
* Following the transfer, don't forget to return
* the modem to the idle state if necessary.
*

* Modem checked, number dialed (Y/N)?

```

**NOTE**

Phone numbers entered must conform to both the requirements of the modem in use (outlined in the corresponding modem user's guide), and the local phone system, whether PBX or local PTT. The number should be called by a person to verify first that it can be done, and second that the access method can be entered according to specific modem requirements (For example, in a PBX system where a nine (9) is required to dial on an outside line, the Support Link I modem would require the following format : 9k5551234).



Proactive Data Communications

After the dialing is complete the following messages should appear in the job listing:

```
**** Answer Tone ****
**** On Line ****
**** Start Remote Logon ****
**** Remote Logon Complete ****
```

Once "ON LINE", Dial speed-senses on the remote system. When a colon is received from the Response Center system, the logon string is transferred to the remote. Dial then waits for the remote : to return indicating that logon has succeeded. Once logon is completed, Dial terminates and the Xfer procedure takes over to transfer the file.

### Xfer Procedure

The Xfer portion of PMNXFRDP performs the following functions:

- o Sets up the port for binary transfers at 1200 baud.
- o Sends direction and file creation information.
- o Transfers the file.

The port is set up to allow binary transfers at 1200 BAUD on both the customer's system and the Response Center's system as follows:

```
Input speed - 1200 Baud.
Output speed - 1200 Baud.
Terminal type - 18.
Echo - Off.
Term character - DC1.
Parity - Even.
Parity check - On.
Turn off CR/LF after reads.
```


Next, the Response Center is informed that it will be receiving the file, and all information necessary to create that file is sent in an initial packet. Once this packet is acknowledged by the Response Center software, the operator is informed of the transfer by the following message to the console:

```
**** PMNXFRDP program initiated ****
**** PMNXFRDP transferring out file TFILE.HP35136A ****
(HPTrend case)
```

At this point the data is transferred record by record. Alternating blocks of data are sent and acknowledged by alternating ACK0's and ACK1's to keep track of block transferred. If an error occurs, a NACK is sent and the packet is resent. At the end of the transfer, a 'BYE' is executed after the program finishes on the remote system to ensure session logoff.

Many error conditions can occur on an unconditioned phone line. Fortunately, PMNXFRDP can inform the customer by printing the abort reason on \$STDLIST. Trivial errors such as failed speed sensing, will be retried two times per contact attempt in case of bad timing.

Xfer uses one "trick" to avoid data overruns from the remote system. The HP 3000 is an echoplex system, ie. it is either reading or writing to the terminal but never both at the same time. Applications such as PMNXFRDP must be careful that the read is started before the remote starts to send data, not easy on a heavily CPU bound system. Fortunately, the HP 3000 emits a DC1

	HP 3000	SM05/29
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Proactive Data Communications

character at the beginning of every read (certain sub-types only). As long as each system in the Xfer process terminate reads on this DC1 character, no overruns can occur.

As the file is transferred, a "heartbeat" of the transfer is printed to the JREDUCE job listing. An example of a successful transfer follows:

```
**** Starting file transfer ---> 25 total records ****
**** 15 Records transferred 60% complete ****
**** File transfer completed ****
**** Disconnecting from remote system ****
**** PMNFRDP program terminating ****
```





## HP RESPONSE CENTER OPERATIONS:

Although of no direct concern to customers, there are some interesting aspects of the HP Response Center sides of HPTrend and Predictive Support. Both are highly integrated and automated. For example, Predictive Support uses an expert system as part of its event analysis system. Highlights of the systems are described in the remainder of this section.

Please note that due to datacomm limitations, certain countries do not take advantage of the capabilities outlined in this section.

### Predictive Support at the RC:

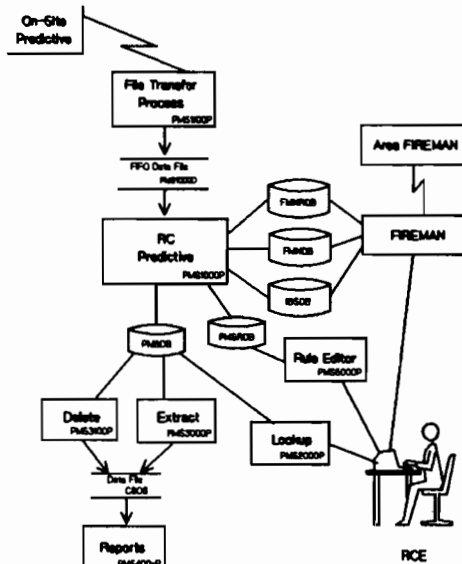



Figure 4

Figure 4 shows the flow of data as it is received and processed at the Response Center (RC). Data is received by the datacomm process (described previously) and stored in a message file. The main RC Predictive process reads and processes the events in the following sequence.

- \* Customer information from the HP support contract tracking system (also known as the Installed Base System, IBS) is retrieved and added to the information received from the message file.
- \* This data is stored in the Predictive Maintenance System Database (PMSDB) allowing access by system or by individual unit on the system
- \* An inference engine in the main program uses rules contained in the rule base (PMSRDB) to determine if an RCE needs to analyze this call. If so, the HP Field Resource Management system (FIREMAN) is notified that a Response Center Engineer (RCE) must look at this system.
- \* Processing statistics are added to PMSDB including instructions generated by the inference engine.

	<i>HP 3000</i>	<b>SM05/31</b>
	<i>INTERNATIONAL CONFERENCE</i>	
	<i>VIENNA 1987</i>	

**Response Center Predictive**

The RCE's main role is to receive requests for Predictive analysis from the FIREMAN system, look at the information in PMSDB, determine the appropriate action to take, and make notations in PMSDB. Many of these notations will be used later by the inference engine in making subsequent decisions regarding the particular site. To alert the Customer Engineer of the problem, the RCE presses a soft-key and FIREMAN automatically routes the call to the appropriate field engineer.

## HPTrend at the RC

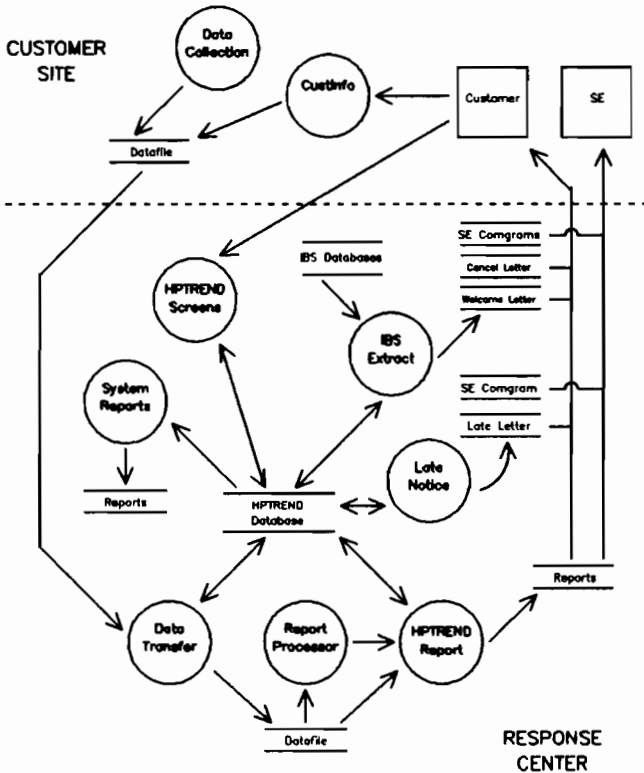


Figure 5

Figure 5 shows a similar data flow for the RC side of HPTrend. The routine operation of RC HPTrend requires no human intervention (except for the stuffing of reports and letters into envelopes). HPTrend is notified of valid HPTrend subscribers via IBS. When a system calls in to the RC, it is checked against the IBS data to verify that it is authorized to print a report. Additionally, the system is checked to make sure that it is the correct time for the transmission. If not, the system is rejected and informed as to why. If a system does not call in at the expected time, a late notice is generated. Once the system is approved, the data is accepted, the HP account team data is added to the report file, a report processor creates the report, and the report is printed via laser. The output is usually mailed out to the customer and HP account rep the day following reception of the customer data.



## NEXT RELEASE FEATURES:

### Predictive Support A.02

The current version of Predictive, A.01, will be replaced by the enhanced A.02 release this year. A.02 will contain most of the following features:

- \* New disc products will be supported
- \* Existing rulebase will be updated/adjusted
- \* System wide vv.uu.ff's and checksums will be monitored and changes reported to the Response Center
- \* New File System Verifier utility will be incorporated
- \* Most rule updates will be automatically downloaded to customer sites via the A.02 datacomm and new Response Center software

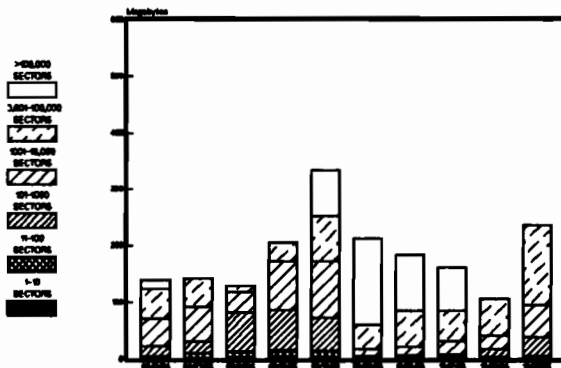
### HPTrend UB Delta 2 MIT

The version of HPTrend released on the MPE UB Delta 2 MIT will provide three new graphs as listed below:

- \* Daily terminal read transaction rate
- \* Hourly terminal read transaction rate
- \* Daily disc free space quantity (example graph below)



Disc Free Space




## Acknowledgments:

The section regarding Onsite Predictive was largely extracted from the article "Predictive Support: Anticipating Customer Hardware Failures", from the November, 1986 issue of the *Hewlett Packard Journal* written by Dave Wasmuth and Bruce Richards.

The Onsite HPTrend section was produced from HP internal documentation for HPTrend written by Zahid Khan and Gerry Wade.

The Proactive Data Communication section was written by Russell Lown.

This paper was written specifically for the Vienna Interex meeting held in March, 1987. It was prepared and presented by Bruce Richards, a section manager of Hewlett Packard's Knowledge Systems Lab in Cupertino, California.

	HP 3000	SM06/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## IS ONLINE BACKUP POSSIBLE OUTSIDE SPECTRUM ?

Joerg Groessler  
Joerg Groessler GmbH  
Rheinstrasse 24

1000 Berlin 41 West Germany

### Overview

Until today users of the HP3000 are requested to stop their daily work whenever a partial or full backup is performed. With the Spectrum program HP has announced an online backup facility which probably will reduce the downtime caused by backup to almost zero. This facility, however, will not be available to the current HP3000 customers. This presentation will explain two basic approaches to an online backup system in MPE.

### What is ONLINE BACKUP ?

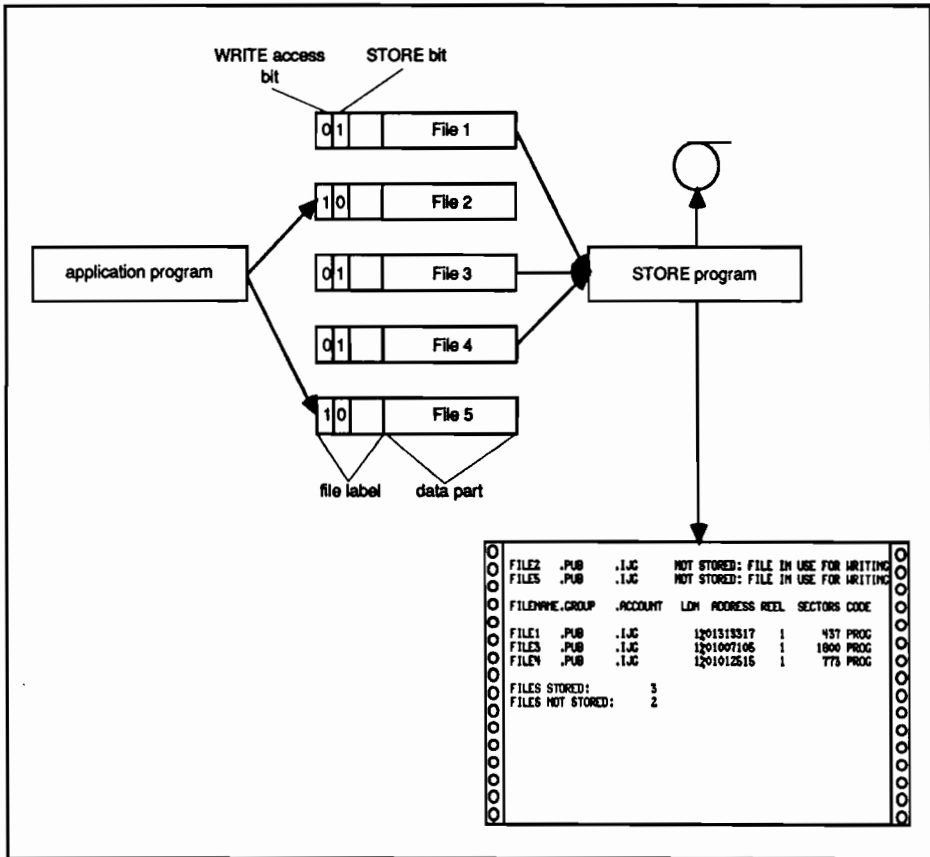
Using the existing STORE facilities (HP's STORE or IJG's BACKUP/3000) the users are required to close files which have been previously opened for write access.

Reason:

- Files cannot be stored in 'zero time'.
- Data which will be written to files will be stored if the file or this part of the file has not yet been stored.
- Some parts of the STORE tape contain more actual data than other parts.

Result:

- Files which are marked as being 'opened for writing' will not be stored.
- A 'STORE bit' is set in the file label to prevent files which are candidates for STORE from being opened for writing.



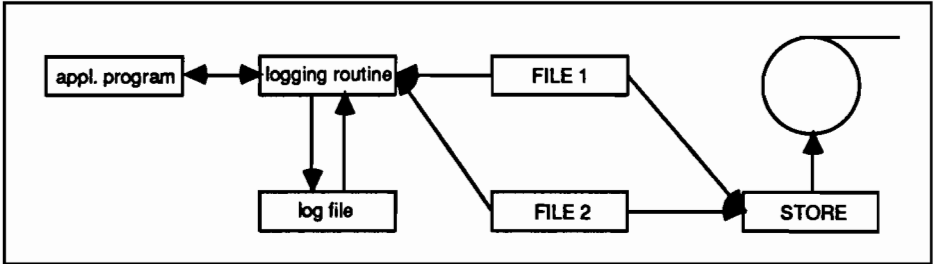
In an ONLINE BACKUP users have to close their files only once at the beginning of the STORE process to put the file system into a defined status. During backup these files can be opened again, even for write access.

To ensure the integrity of the file system the write requests performed during the backup have to be handled differently than usual. This is done with a special logging routine which is called before the actual file write is performed.



## **First Approach: Actual File I/O Logging**

The next picture illustrates the principle of online backup using actual file I/O logging:



- all write requests are performed only into the log file rather than the actual user file.
- read requests by the user program have to check the log file whether this part of the file has already been logged (in that case the data has to be read from the log file rather than the actual user file).
- The STORE program simply stores the original files since they are not changed during backup.
- After the end of the backup the files have to be actualized by the contents of the log file.

### **Advantages:**

- Minimum overhead for all write requests (one write request to the actual file results in one write request to the log file).
- No changes to the STORE program.

### **Disadvantages:**

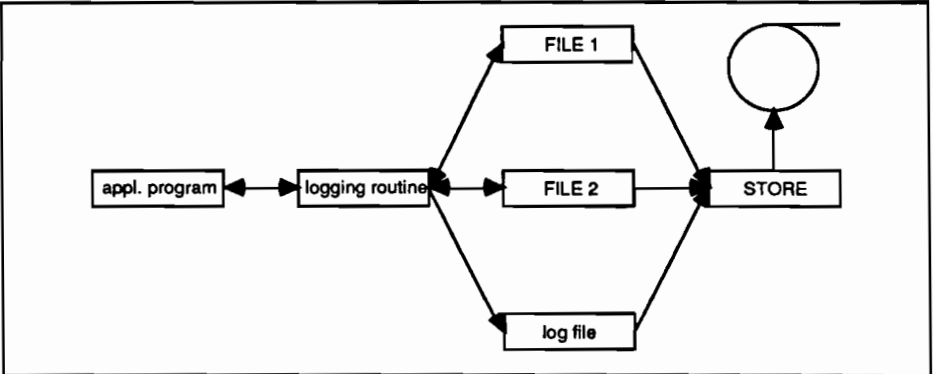
- Overhead for read requests by application programs increases with the number of records already logged and can be very high.
- Hard to recover after a system halt (the files have an old status and the log file might have been destroyed).
- Hard to debug.



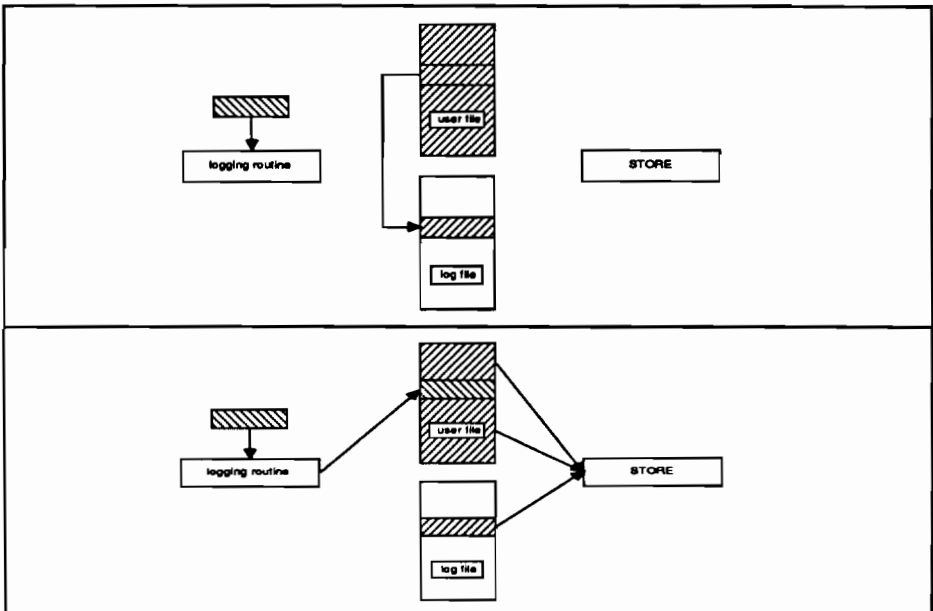


## Second Approach: Reverse Logging

The next picture shows the principle of 'reverse logging':



- Before a write request is performed the 'old record' is read from the file and written to the log file.
- Each write request has to check if copying the old record is really necessary or if it has been done before.
- The STORE program has to collect its file data out of the actual data file (if this part has not been affected) or out of the log file.



**Advantages:**

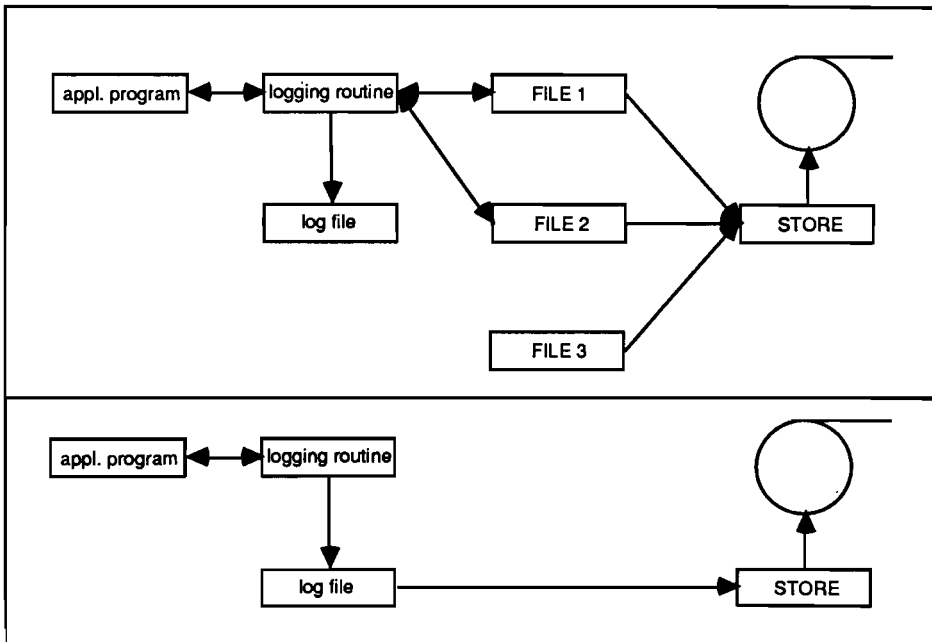
- Much more secure than first approach.
- No overhead for file read requests (they can always be done using the actual file data).
- No additional updating after the end of the backup.

**Disadvantages:**

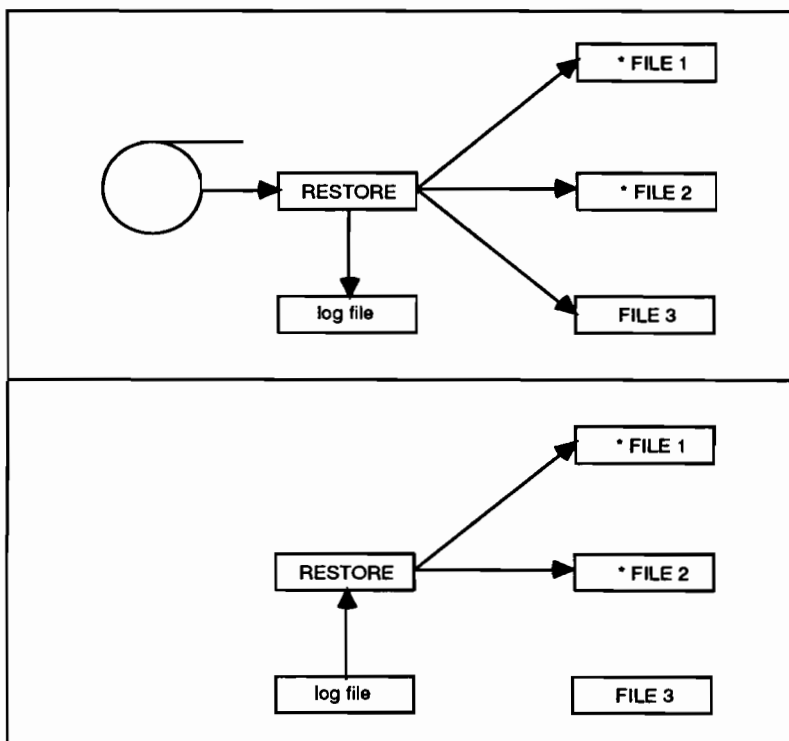
- Overhead for write requests increase with the number of records in the log file and can be still high (but not as high as in the first approach).
- STORE program has to check the log file for 'old records'.

**Third Approach: Store LOG Files**

Another approach does not change the way files are been written during the backup procedure. However write requests are written to the log file and the log file is appended to the end of the tape after all files have been written to tape:



RESTORE creates files which have been open for writing during STORE in an inconsistent status. Using the LOG file all write requests are redone to ensure file integrity:



### Advantages:

- Minimum overhead for STORE operation. One additional disc I/O per 50 to 100 'real' disc I/Os.
- High security since file I/O is not affected by logging
- Better to debug since LOG files are stored on tape for further analysis
- Synchronization point at the end of backup rather than at the beginning (better for users who normally do their backup in the afternoon).
- Possibility to 'spread' LOG files on STORE tape in case 'out of disc space'.

### Disadvantages:

- RESTORE operation more complex than on other approaches (RESTORE however happens much less often than STORE)



## Application Recovery

### Automatic Restart of an Application after System-Failures

(Eduard Stiefel, SWS SoftWare System AG, CH-3007 Bern)

#### 1. The Problem

Whenever your DP-system or your program aborts during a transaction, an organized restart - a recovery - is necessary to reconstruct the consistency. A simple new-start would mean that you are working on with inconsistent data.

By the term transaction we understand in the following context a series of logically connected actions on data, for example in an IMAGE database. With action we mean the adding, changing and deleting of data.

#### For example:

In a financial accounting the transaction "book-keeping" (for example) consists of the holding of the book-keeping record, an update of the concerned accounts and the subsequent update of the balance. After the beginning of the transaction the data will not be consistent, before the transaction is closed. An abnormal program termination inbetween would leave the database in an inconsistent condition; for example debit-booking finished, but the credit one still open.

How could this problem be solved?

#### 2. Restart with general tools (System Recovery)

A first method to correct the database uses the help-facilities and tools, which are available on the system. Since these tools, which belong to the system and which could be from HP or a third party, must be applicable in any case, we call this method system recovery.

In case of an IMAGE-database the process runs as follows:

a) You make a DBRESTOR or a RESTORE of the last copy of the concerned database, then you perform with DBRECOV the intermediate transactions and stop just before the critical transaction.

or

b) You use the Roll-Back-Method of TurboIMAGE to exclude the last transaction(s) of the database.

If your application uses KSAM- or MPE-files too, you probably don't have system-tools to reconstruct the consistency. Since most applications use IMAGE, this method seems to be an easy, but perhaps time-consuming way.

In this special case the problem is that the system-tools are general tools, which of course do not know the specifics of the application. That means that such a restart could only be done by a person, who knows the database and their structure as well as the application or the concerned program.

Is such a person always available? How long does such a recovery take? Can you always be sure that the program has normally been ended and that a restart is not necessary?

In how many of these cases it is being worked on without a recovery, and just later some one is noticing an error. If you work with distributed systems or third-party application software, such a situation could be much more critical.

### **3. Restart by the application itself (application recovery)**

If you are conscious of all these problems, the question arises, if it is possible to design an application-package with an integrated recovery-approach.

#### **Requirements:**

The design of an application should guarantee besides all other aspects - the automatic restart after program aborts or system failures in such a way that there is no inconsistency of data.

In the following we will try to find a way to fulfill these requirements in a multi-user-environment.

1. A system-failure (not a program-abort) could also cause IMAGE-internal inconsistencies, (e.g. broken chains), which cannot be removed by an application-recovery. The use of ILR (intrinsic level recovery) leads automatically to a reconstruction of the internal consistency at the first DBOPEN after the restart.
2. If your database gets lost (e.g. by damage of your disc), a system-recovery is the only applicable method.
3. A transaction consisting of only one DB-action is not critical. A logical inconsistency cannot occur. This concerns most of the masterfile - update - programs.
4. So the critical ones are the interactive and batch-programs, with transactions consisting of more than one database-action.

Since such a Batch-program usually runs during the night and in case of a program-abort the restart with the system recovery method is relatively simple, we will restrict ourselves to the interactive programs.

If in case of a restart an unfinished transaction should be realized, the initial situation must be kept in mind, and all actions must somehow be marked as belonging to this transaction. The end of the transaction must somehow be recognizable.

In order to reach this in a multi-user-environment, the following proposal seems to be a reasonable way. During the interactive retrieval and updating of data only one process executes the necessary transaction in the database. The corresponding program, let us call it the main program, runs permanently or at certain times as a batch-program. All other programs do only database inquiries however all updates are not executed by themselves, but transferred to the main program for execution.

If only a single process executes the database-actions, this process could keep the initial situation of a transaction in mind (e.g. in a help-file), it could also execute the transaction deleting the help-file afterwards and continuing with the next transaction. If this main program meets a clean situation after the program-start,

it is able to realize and manage an aborted transaction. The same principle is used by IMAGE concerning ILR.

How does the main program get to its informations?

The other programs send their informations either in a message-file, from which they can be reread or they write them temporarily in a database and transmit via message-file just the processing-command. Important is that the single write-operation comprises the whole transaction; otherwise we have a logical inconsistency again.

In the following example a concept of an automatic restart will be outlined.

**Example:**

In the initially mentioned financial accounting a booking will be keyed in with an entry program on the screen and written in a file at whole. Either subsequent or for some bookings together the booking-process is started.

The main program gets a transaction number, keeps the initional situation in mind and processes step by step the DB-operations, which it marked with the transaction number. At the end the stored inital situation is deleted, the database is consistent again.

If a system- or a program-abort occurs, the main program can guarantee the clean restart, an additional, human intervention is not necessary for the recovery.

Adress of the author:

Eduard Stiefel

c/o SWS SoftWare Systems AG

Schönauweg 8

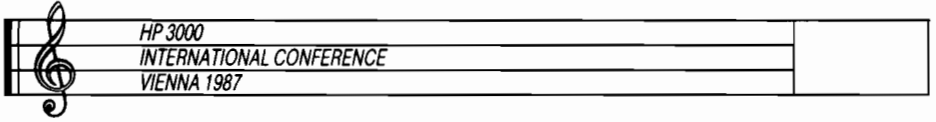
CH-3007 Bern

or


c/o SWS SoftWare Systems GmbH

Postfach 1710

D-7858 Weil am Rhein



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

	HP 3000	VS01/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Chemserv Consulting  
Gesellschaft mbH  
St. Peter Straße 25  
A-4020 Linz / AUSTRIA

" C C - S T I P R O "

=====

INVENTORY: Stocktaking by Random Samples

- PAYS FOR ITSELF -


by Josef Angerer

1. PRELIMINARY REMARKS

"CC-STIPRO" is a software package which enables stocktaking operations to be conducted on a random sample basis. The system has been designed to facilitate the time-consuming and expensive business of labour-intensive stocktaking as part of the annual inventory procedure.

The value of stock held can be calculated using a number of well-known methods that satisfy legal requirements. Recent legislation however, has opened up a further possibility of using recognised mathematical and statistical operations in random sampling to ascertain the levels of capital goods according to type, quantity and value.



	HP 3000	<b>VS01/2</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

In order to perform stocktaking operations on a random sample basis, two conditions must be fulfilled.

- The number of items in stock must be sufficiently large.
- The stocktaking must be accurate in terms of type, quantity and value, must be computerized and must comply with the basic standards of regular accountancy.


## 2. REQUIREMENTS

### - COMPREHENSIVENESS

All stock items must be included without exception in order to satisfy the prerequisites of random sampling. In other words, the existing bookkeeping system must reliably reflect the stock situation in its entirety.

### - ACCURACY

Accuracy in stocktaking is a somewhat relative term, since even an overall survey of stock during a conventional inventory (whether continuous or just on the stocktaking day) must assume an error margin of up to 2 % against the true value of stock. At times, this discrepancy may be even greater. In the case of stocktaking by random samp-

	HP 3000	VS01/3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

les, the relative margin for random sample error lies within 1 % of the true value of total stock, meaning that overall accuracy can be defined within these two limits.


- VERIFIABILITY

It must always be possible to verify which capital goods according to type, quantity and value are comprised in the random sample.

Stocktaking by random samples is a mathematical-statistical operation which produces an estimate for the true value of stock held. The estimate itself is calculated by using the generative mean procedure.

For the sake of definition, the "true value of stock held" is the total value of all items in stock which have been counted accurately and extensively; the "estimate" is the projected value of the stock held and is based on a random sample of a small selection of items in stock.

Usually, these two values do not coincide with each other. But the estimate can be determined so that it does not diverge more than "X" percent from the true value of stock held with a predefined probability. The accuracy of the

	HP 3000	VS01/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

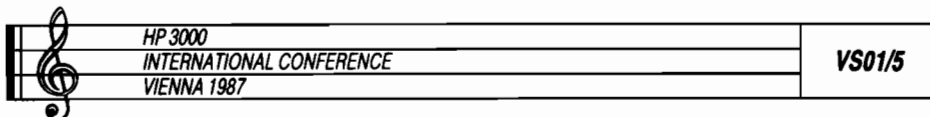
estimate (expressed as a percentage) is known as indicative probability.

To use an example:

A warehouse has a total book value of 8 000 000. The task is to find an estimate for the true value of the stock held by using STIPRO. This estimate must have an indicative probability of 95.5 % that it diverges no more than 1 % (80,000) from the true value of the stock held (= permissible error, random error).

CC-STIPRO provides an estimate, ie projected estimate value of 7 800 000. In other words, 95.5 times out of 100, the true value of stock held is between 7 720 000 and 7 880 000. Since the true value of stock held will not diverge much from the book value, the maximum permissible divergence is often regarded as relative to the total book value. 80 000 is 1 % of 8 000 000, and the level of divergence is known as the range of reliability (in this case 1 %). The degree of certainty is defined by the factor: degree 2 of certainty corresponds to an indicative probability of 95.5 %.

A 95.5 % degree of certainty means that for 95.5 % of 100 inventory operations, the CC-STIPRO value of stock held di-



verges no more than 1 % of the true value of stock. This corresponds to the relative margin for error when taking random samples.

### 3. In conclusion

During the operation stocktaking by random samples an estimate is made of the true value of stock held.

The accuracy and reliability of this estimate is expressed in the degree of certainty and range of reliability.

The number of random samples taken depends not only on the number of items held in stock and the warehouse structure but also on the degree of certainty and range of reliability which is required.

Fundamentally, each and every item in stock must be able to be included in a random sample.

Special features: COBOL

VIEW/3000

Serial Files

Forget the time-consuming and expensive business of stock-taking in the future. CC-STIPRO is the ultimate and modern method to settle an imposed liability.



HP 3000

INTERNATIONAL CONFERENCE

VIENNA 1987



## THE INTEGRATION OF HARDWARE AND SOFTWARE MAINTENANCE

Judy Hayner  
HEWLETT-PACKARD

### ABSTRACT

The system maintenance process is evolving from the traditional hardware and software focus to a stronger total system support orientation. This paper will examine and highlight these trends in the support arena. Key issues considered will include the extent to which hardware and software maintenance services, techniques, and delivery methods will potentially merge and change, and more importantly, the implications of these changes for system users and vendors.

Many factors are influencing this move toward hardware and software maintenance integration. The most important of these include:


- the growing emphasis on remote support tools and capabilities in both hardware and software support to reduce the time and costs associated with problem diagnosis and resolution;
- the proliferation of networking and multiple system environments where a blend of hardware and software expertise is essential;
- the changing methods and processes for hardware and software installation, system and application software updating, and problem escalation management;
- the growing customer demand for more support resources devoted to the implementation of software applications.

The effect of these trends will be considered from the perspectives of both system users and system vendors.

### INTRODUCTION

The importance of high quality support, throughout the life cycle of hardware and software products, is growing dramatically due to its significant influence on the system users' overall success and satisfaction. Contributing to this growth are emerging factors such as a greater variation in the expertise level of users, more complex operating environments, and the use of systems for more demanding and time-critical applications.

As users' needs change over time, the methods of delivering support will evolve to more closely match those needs. System

	HP 3000	VS02/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

vendors must strive to increase and maximize their productivity, efficiency, and effectiveness in delivering support services. They must endeavor to provide services which embody the qualities that customers value most: consistency, continuity, responsiveness, timeliness, flexibility, and cooperation.

The quality of the support provided by the vendor has a major impact on a customer's ability to meet business objectives. High quality is necessary for a fruitful working partnership to exist between the vendor and the System Manager or Operator. Providing this level of support, and establishing and maintaining a productive relationship with the customer, must be the vendor's goal.

This paper outlines some of the ways in which support delivery methods and techniques are changing to achieve the improvements and benefits described above. The primary focus is on hardware and software maintenance activities, however, it will become apparent that the changes in this area greatly affect a vendor's performance in all other stages of support. The potential change in service offerings is also examined. And finally, consideration is given to the implications for the system user and system vendors.

## THE DELIVERY OF HARDWARE AND SOFTWARE MAINTENANCE

Traditionally, many vendors, including HP, have provided hardware and software support through different field organizations. This split between hardware and software expertise is beginning to disappear. The responsibilities of the field organizations are evolving toward more system oriented specializations according to their established skills and strengths. The development of new support technologies and tools is also influencing this movement, through the increasing ability to centralize and streamline many of the maintenance activities. The benefits of these changes will become clear as the emerging roles of each of the support entities are examined individually.

### Central Support

One major trend has been the centralization of telephone assistance to aid customers in system usage and problem resolution. At Hewlett-Packard, this has been done in the Response Center Organization. This centralization enables the use of shared databases including the most up-to-date information, remote support tools and diagnostic capabilities, and specialized teams of engineers with breadth and depth of knowledge. A very large percentage of all software related problems can now be solved remotely. The need for on-site assistance is rapidly diminishing. The benefit to the user is more timely, effective, and economical problem resolution, as well as a consistent and efficient interface.

These support centers provide the first point of contact for customers, regardless of the question, and can help to distinguish between hardware and software problems. If more assistance is needed outside the center to resolve a problem, the initial information which has already been gathered will greatly speed the process. In addition to reacting to customer requests, proactive services, such as predictive maintenance, can be effectively provided from a central support organization. System users will continue to see expanded service offerings from support centers.

### **Field Maintenance Support**

In the few cases that on-site assistance is required to resolve a problem, an engineer in the field will provide this assistance. The organization historically focused on hardware maintenance and product availability, the Customer Engineering Organization (CEO) at HP, is beginning to also take on the responsibility for on-site software problem diagnosis. The strengths of this organization, in rapidly responding to customers' on-site needs, make this a logical alignment of activities.


An additional activity that leverages the skills of this evolving system maintenance field organization is on-site software installation. The same engineer that provides hardware installation will install the fundamental operating software on new systems requiring this assistance. However, the need for on-site installation assistance for software add-ons and updates will be minimal in the future. (The reasons for this are outlined below.)

As one support organization begins to provide on-site assistance in the installation and maintenance of both hardware and software products, its knowledge of software will broaden, enabling a more system oriented focus to maintenance. This blend of expertise prepares this organization for the responsibility of network installation and troubleshooting. The growing number of customers implementing sophisticated networks demands a greater attention to developing extensive network support capabilities. A key step in this process is the alignment of network support responsibilities within the organizations best suited to provide them.

### **Self-Support**

The System Manager is also beginning to take on a greater role in software maintenance. With the help of technology advances and improved software quality, this capability will continue to grow. The installation process will be straight-forward and simple, allowing the System Manager to complete it with little or no assistance. The vendor's ability to create and distribute customized software updates, and the availability of telephone assistance, will both greatly reduce the need for on-site update installation assistance. Even for new systems, the trend is toward pre-installed operating systems, thus minimizing the need for any installation assistance.



	HP 3000	<b>VS02/4</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

System Managers and users will also begin to have on-line access to vendors' support databases. This will enable them to play a more active part in problem identification, thus reducing the total resolution time.

### **Escalation Support**

A streamlined escalation process, in the case of an unresolved problem, is essential. To address this need, Escalation Managers are placed in a position enabling them to draw upon any resources needed to resolve a problem. These managers take responsibility for all escalated problems, hardware and software, and in many cases, will elicit the on-site assistance of the field maintenance organization.

### **Planning and Implementation**

The realignment of roles and activities described above is not only a major stride toward improving the software maintenance process, but it also significantly impacts the delivery of planning and implementation services. The field organization which has historically provided all on-site software support, the Applications Engineering Organization (AEO) at HP, is able to focus its efforts on helping customers identify, plan, and implement total product, system, and support solutions. This assistance includes activities such as pre-sales planning, education, implementation assistance, account management, consulting, network design, performance analysis, and the management of custom projects. The demand for these services is one of the fastest growing in the support industry, and this field organization has the proven strength to successfully provide them. By separating the proactive services from the more reactive ones, both field organizations are able to more effectively meet customer needs.

### **SERVICE PACKAGING**

Support service offerings differ substantially from vendor to vendor depending on the overall service marketing strategy. Some vendors bundle support services with the products and others offer varying degrees of flexibility in purchasing support. One point is clear, there does not seem to be, or need to be, a direct correlation between service delivery (who delivers support) and the service packaging (how customers obtain support).

The packaging and marketing of support must take into consideration both similar and unique needs across the customer base. The trends point toward simplicity, yet flexibility in service offerings. The integration of hardware and software maintenance into one set of services is not necessary, but the presentation of the total support solution is. The opportunities for improvement are in helping customers define their support needs, in understanding how they can be met, and in making the administrative processes of delivering support smooth and easy.

## SUMMARY/IMPLICATIONS

The delivery of maintenance services is evolving for increased productivity and customer satisfaction. Factors influencing this evolution include advances in support technologies and tools, as well as changing customer needs and environments. For efficient and cost-effective delivery, the following elements will emerge:

- a centralized support organization where a high percentage of problems are resolved remotely;
- a field organization focused on system maintenance, including hardware maintenance, on-site installation assistance, and on-site software and network problem identification as needed;
- a second field organization focused on the planning and implementation of complete system and support solutions;
- a growing participation in software installation and maintenance by System Managers/Operators and users;
- a well defined and powerful escalation process for addressing unresolved problems.

For system vendors, the operational changes described indicate the need for some reallocation of resources and shifting of responsibilities. This requires expanded training efforts and a well planned, cooperative implementation. The fact that each organization will focus more clearly on their previously developed strengths and areas of proven success will ease the transition.

The key to a successful evolution is for each organization or individual to begin taking on new responsibilities without totally releasing the old. For a period of time, responsibilities will be jointly owned to ensure a transparent transition. In this way, only the improvements in the process will be detected by customers.

For System Managers/Operators and users, the benefits will be significant: a more effective initial point of contact, faster response when on-site assistance is required, and more individualized and expert attention for planning and implementation purposes. Of key importance in taking advantage of the improvements will be the early planning of all support needs for the life of the system. This initial partnership with the vendor will then continue to contribute to the customers' success and satisfaction with their system and support services over time.

-----


*Judy Hayner is the Product Marketing Manager at the Product Support Division in Cupertino. She has been with Hewlett-Packard in the marketing of support services since 1980. She holds a BA degree from Stanford University in Environmental Design and an MBA degree from the University of California at Los Angeles.*



*HP 3000*

*INTERNATIONAL CONFERENCE*

*VIENNA 1987*

	HP 3000	VS03/1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

RINs, RINs, RINs  
Benedict G. Bruno  
S.T.R. Software Company  
P. O. Box 12506  
Arlington, VA 22209

## Introduction

Several articles have been written discussing the merits of the HP 3000 Computer System as a transaction processing oriented machine. Topics have included process handling, interactive on-line systems, etc. An excellent resource available to the standard MPE user/programmer for process handling control is the Resource Identification Number, better known as a RIN.

RINs come in two flavors. Local RINs are available only to the creating session or job. Global RINs are permanently saved on the system disc and are available to anyone on the system.

The concept of RINs has not been adequately described. This paper will discuss the benefits of local versus global RINs in a multi-processing environment, i.e., how may RINs be utilized for the control and monitoring of user applications executing on the HP 3000. RINs are extremely effective in the process handling environment. Several programming examples will be provided.

Although MPE provides several system intrinsics and commands for creation and access of local and global RINs, no software tool is provided for display of this information. Thus it is difficult, almost impossible, to display the RIN number, password, creating user/account, information of locking process, and information of waiting process. This information is maintained by MPE in the System RIN table.

The contents of the System RIN table have been formatted for the user in the supplied SHOWRIN program. The author has developed this program for both MPE IV and MPE V based systems. Depending upon capability, portions of the RIN table are formatted in displays similar to the HP On-line Performance Tool (OPT) program. Using this SHOWRIN program and the discussion of RIN usage, you can start integrating local and global RINs into your HP 3000 applications right away!

## Local RINs

MPE provides local and global RINs for process control. In order to begin our discussion, let us define local and global RINs.

Local RINs can be described having the following attributes:

- \* Created, accessed, and released on a session or job basis only. (intra-job)
- \* Programmatic MPE intrinsic access only.
- \* User maintains control of RINs by specifying which task each local RIN number is associated.
- \* The MPE LOCRINOWNER intrinsic returns the process identification number (PIN) of the process currently locking the local rin.

A local RIN is a resource allocated on a job or session basis for use by any process executed during this job or session. Local RINs are created and released only using the GETLOCRIN and FREELOCRIN intrinsics programmatically. The GETLOCRIN intrinsic allocates the requested number of local RINs for the current job or session. The FREELOCRIN intrinsic releases all local RINs previously obtained with the GETLOCRIN intrinsic.


Local RINs are accessed using the LOCKLOCRIN and UNLOCKLOCRIN intrinsics. Local RINs are referenced with the integer value assigned within range of the number allocated with the GETLOCRIN intrinsic. Thus, if the user requests 24 local RINs with the GETLOCRIN intrinsic, then the LOCKLOCRIN intrinsic may lock RIN numbers one through 24. The UNLOCKLOCRIN intrinsic performs the same way.

The MPE LOCRINOWNER intrinsic provides the PIN of the locking process for the supplied local RIN. This capability prevents deadlocks and informs any process in a process tree of the currently locking process. Thus, there is no need for a display of the local RIN entries from the system RIN table.

### Global RINs

Global RINs can be described having the following attributes:

- \* Created and released on a system wide (global) basis for any session or job on the system. (inter-job)
- \* User assigned RIN password is associated with a unique entry in the system RIN table using the GETRIN MPE command. Released only by creating owner with the FREERIN MPE command.
- \* System RIN table is system disc resident on logical device number 1. Modified only during reload.

	HP 3000	VS03/3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

- \* Any job or session may access the global RIN using the LOCKGLORIN and UNLOCKGLORIN intrinsics by supplying the RIN password and RIN number.
- \* The HP supplied SYSDUMP program displays the global RIN number with the creating user and account. The user specified password is NOT displayed. Hence, the need for the SHOWRIN utility program.

Global RINs are available to any session or job on the system provided the number and password are known. Global RINs are assigned to a creating user and account with the GETRIN command. The RIN associated with this password is permanently maintained in the System RIN table which is system disc resident. An entry may be removed using the FREERIN command. Note that the RIN entry may only be removed if the creating user and account match the logged on user and account names!

The size of the global RIN table residing on the system disc is configured with the SYSDUMP dialogue. Changing this size can only occur during a RELOAD. The standard MPE configuration is delivered with a table size of 48 entries. In order to utilize the global RIN feature, you should increase the size of the table appropriately.

The following commands can be executed to create and free the global RIN assigned using the password of GRIN.


```
:HELLO user.account
:GETRIN GRIN
RIN: nnn

:FREERIN nnn
```

MPE responds with the integer value assigned to the RIN password with the GETRIN command. It is important that you always remember this association. Currently, no HP supplied utility displays the RIN entry number, password, and creating user and account names.

You should note that the FREELORIN intrinsic removes all created local RINs for the current session or job. The FREERIN MPE command releases ONE global RIN from the system RIN table if currently signed on as the creating user and account.

In order to continue, the reader should now note that the local RINs may be very effective during process control within an individual process tree, i.e., intra-job level processes. Global RINs are very effective during process control within several processes executing from different jobs, sessions, and accounts, i.e., inter-job level.

	HP 3000	VS03/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

### Global RIN Example

Global RINs are valuable in the monitor and control of several programs comprising an application. By assigning a global RIN to each program in the application and requiring each program in the application to lock its associated RIN, we are guaranteed that only one copy of the program is executing. Some programmers implement this feature by opening a file for exclusive access. Since the global RIN is locked during the entire execution of this program and additional data structures may need to be locked, i.e., IMAGE/3000 data bases, KSAM and MPE files, etc., the program file must be :PREPped with MR (multiple resource or better known as multiple RIN) capability in order to hold multiple locks. The user, group, and account capabilities must also be modified for MR capability.

Once each of the programs in the application is initiated and the lock is maintained on their RINs, another monitor program may control the execution of these processes even further. Suppose that this monitor program is responsible for the initiation, control, and termination of all programs within this application. Once each process is started, the RIN is held for the duration of this process. The monitor need only attempt a conditional lock in order to determine if the process is still executing. If the conditional lock succeeds, then the process is not executing since the monitor process was able to lock the RIN; otherwise, the lock fails and the process is executing since the RIN is unavailable.

This technique is further enhanced by utilizing the writers id feature with the open, close, and data record types of the MPE message file system. The requirement is to have each program in the application open the monitor message file for write access. If the monitor program opens this message file and enables the writers id feature, then as each process opens, writes, or closes this file, the monitor program receives a record containing the writers id, the file access, and data if it is a write. The writers id is an integer assigned by the file system in ascending order to each program opening the file. If a close record is detected unexpectedly, then the monitor program need only identify which program has failed and respond in a controlled manner. The monitor may restart the failing program or may request all other programs in the application to terminate immediately.

In addition to the initiation and termination of programs within the application, the monitor program may also control the execution of each of these programs. Specifically, the monitor program may suspend or resume execution of these programs. Utilizing the message file techniques above, the monitor program issues a conditional lock on a secondary RIN for each process requested to suspend. Once locked, the monitor requests that the process suspend itself by issuing an unconditional lock on this secondary RIN. The process is resumed

execution when the monitor releases the secondary RIN. Hence, an excellent method of global process control!

The SHOWRIN program was developed in order to determine which RINs were currently locked and by which process. Another useful feature is that the password could be associated with the RIN entry itself. This feature saved documenting the RIN entries and their passwords into a journal for later reference.

Now let's begin with an example to better explain these features. An application of one monitor and two programs will be used to demonstrate the global RIN concepts discussed earlier. Sample code will be provided in both SPL and COBOLII.

In Figure 1 below, the monitor program is diagrammed to use the parm value of the :RUN command as the number of the global RIN named 'MONITOR'. The monitor program issues a conditional lock on this global RIN. If successful, the program continues; otherwise, an error message is displayed since some other process currently has this RIN locked (this prevents two copies of the monitor program from executing because another copy is already executing). The monitor program opens three message files: the primary message file named 'M' is opened for read access enabling the writers id feature; the additional message files named 'A' and 'B' are opened for write access in order to communicate with process A and process B.

```
:RUN MONITOR;PARM=rinnumber
```

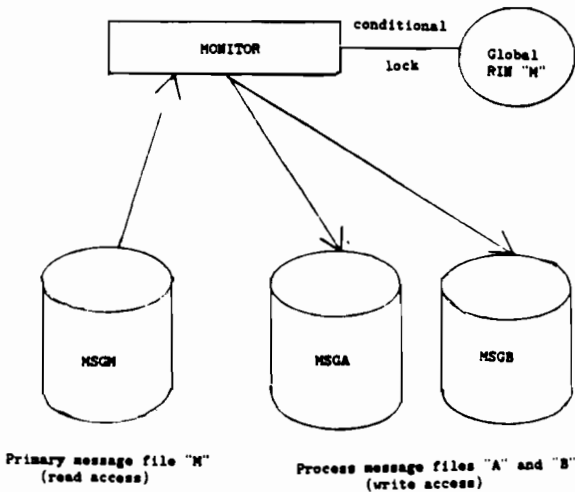


Figure 1: Monitor Program "M" Execution Flow Diagram





The example has been coded in SPL below in Figure 2. The program variable declarations are first displayed. Using the indentation and blank lines provide for easy reading.

Figure 2: Monitor program in SPL

```
1 $CONTROL USLIMIT,MAP
2
3 <<----->>
4 <<** MONITOR Program Version Information. **>>
5 <<** **>>
6 <<** Version Date Who Comments / Description **>>
7 <<** ----- **>>
8 <<** B.01.00 5/31/86 BGB Initial program release. **>>
9 <<** **>>
10 <<** This program will lock the global RIN name of "MONITOR" **>>
11 <<** with the RIN number passed in the ;PARM= parameter. **>>
12 <<----->>
13 $PAGE "***** MONITOR PROGRAM *****"
14 BEGIN
15
16 COMMENT EQUATES AND DEFINES
17 -----;
18
19 DEFINE IR'CONTROL'CODE = LINPUT'RECORD#,
20 IR'PROGRAM'ID = INPUT'RECORD(2)#;
21
22 EQUATE RS = %036,
23 BEL = %007,
24 ESC = %033;
25
26 $PAGE
27 COMMENT LOCAL VARIABLES
28 -----;
29 LOGICAL ARRAY LITERLINE(0:79);
30 BYTE ARRAY TERMLINE(*)=LITERLINE;
31
32 LOGICAL ARRAY LBAL(0:79);
33 BYTE ARRAY BAL(*)=LBAL;
34
35 LOGICAL ARRAY LINFO'STRING(0:127);
36 BYTE ARRAY INFO'STRING(*)=LINFO'STRING;
37
38 LOGICAL ARRAY LINPUT'RECORD(0:119);
39 BYTE ARRAY INPUT'RECORD(*)=LINPUT'RECORD;
40
41 DOUBLE RUN'PARM;
42
43 LOGICAL DONE,
44 LOCK'COND,
45 OK,
46 TIME'2'QUIT,
47 TRUE'COND;
48
49 INTEGER CERROR,
50 CPARM,
51 FERROR,
52 I,
53 IN'LENGTH,
54 INFO'LENGTH,
55 LENGTH,
56 MSGA'FILE,
57 MSGB'FILE,
58 MSGM'FILE,
59 NUM'CHAR,
60 PAUSE'RIN'A,
```

Continuing the program below, the intrinsic declarations are listed in alphabetical order. The GET'INFO procedure will locate the ;PARAM= and ;INFO= parameters of the :RUN statement by locating the terminate stack marker.

Figure 2: Monitor program in SPL (Cont.)

```

61 PAUSE'RIN'D.
62 MONITOR'RIN;
63
64 INTRINSIC ASCII,
65 BINARY,
66 COMMAND,
67 DATELINE,
68 FCHECK,
69 FCLOSE,
70 FCONTROL,
71 FERRMSG,
72 FOPEN,
73 FREAD,
74 FWRITE,
75 LOCKGLORIN,
76 PRINT,
77 PRINTFILEINFO,
78 QUIT,
79 TERMINATE,
80 UNLOCKGLORIN;
81
82 $PAGE
83 PROCEDURE GET'INFO (PARAM, INFOL, INFOSTR);
84 DOUBLE PARAM;
85 INTEGER INFOL;
86 LOGICAL ARRAY INFOSTR;
87
88 BEGIN
89 ..*****
90 <<"" The GET'INFO procedure will locate the ;PARAM= and "">>
91 <<"" ;INFO= parameters from the :RUN statement. This "">>
92 <<"" procedure will correctly execute on any version of"">>
93 <<"" MPE IV and MPE V. Note the Delta-P value change "">>
94 <<"" from %0 of MPE IV to %40000 of MPE V. By reading "">>
95 <<"" stack markers till these values, we eventually "">>
96 <<"" locate the terminate marker and the parameters are"">>
97 <<"" then at SM-4, SM-5, and SM-6. "">>
98 ..*****
99
100 INTEGER QREG = Q;
101
102 INTEGER POINTER SM;
103
104 BYTE POINTER INFO,
105 INFOT;
106
107 ..*****
108 <<"" Start of main code "">>
109 ..*****
110
111 @SM:=@QREG; <<"" Point to current marker "">>
112
113 I:=0;
114 DONE:=FALSE;
115
116 DO
117 BEGIN
118 IF SM(-2) = 0 THEN <<"" MPE IV terminate marker "">>
119 DONE:=TRUE
120 ELSE

```



The FS'ERROR procedure formats a standard error message for any file system error detected while accessing any of the MPE files. This includes the text from FERRMSG and the tombstone from PRINTFILEINFO.

Figure 2: Monitor program in SPL (Cont.)

```

121 BEGIN
122 IF SM(-2) = $40000 THEN <<MPE V terminate marker >>
123 DOWE:=TRUE
124 ELSE
125 BEGIN
126 @SM:=@SM-SM; <<Walk down stack marker >>
127 I:=I+1;
128 IF I = 100 THEN
129 QUIT (-1); <<Cannot find terminate >>
130 END; <<marker so abort! >>
131 END;
132 UNTIL (DOWE);
133
134 PARM:=DOUBLE (SM(-4)); <<SM-4 = value of :PARM >>
135 INFO:=SM (-6); <<SM-6 = length of :INFO >>
136 @INFO:=SM(-5); <<Byte ptr to info string >>
137 @INFOT:=@INFOSTR&ASL(1); <<Byte ptr of target >>
138 MOVE INFOT:=INFO,(INFOL); <<Move to target >>
139 END;
140
141 $PAGE
142 PROCEDURE FS'ERROR (FILE'INDEX, FILE'NUMBER, FS'INTRINSIC);
143 VALUE FILE'INDEX, FS'INTRINSIC;
144 INTEGER FILE'INDEX, FILE'NUMBER, FS'INTRINSIC;
145
146 BEGIN
147 FCHECK (FILE'NUMBER, FERROR);
148 MOVE TERMLINE:="Unexpected FS error ",2;
149 NUM'CHAR:=ASCII (FERROR,10,BAL);
150 MOVE ":BAL,(NUM'CHAR),2;
151 MOVE ": " has occurred on ",2;
152 CASE FILE'INDEX OF
153 BEGIN
154 <<0>> ;
155 <<1>> MOVE " : "MSGM",2;
156 <<2>> MOVE " : "MSGA",2;
157 <<3>> MOVE " : "MSGB",2;
158 END; <<END OF CASE >>
159 MOVE " : " during ",2;
160 CASE FS'INTRINSIC OF
161 BEGIN
162 <<0>> ;
163 <<1>> MOVE " : "FOPEN",2;
164 <<2>> MOVE " : "FCLOSE",2;
165 <<3>> MOVE " : "FREAD",2;
166 <<4>> MOVE " : "FWRITE",2;
167 <<5>> MOVE " : "FCONTROL",2;
168 END; <<END OF CASE >>
169 MOVE " : " ",2;
170 LENGTH:=TOS-LOGICAL (@TERMLINE);
171 PRINT (LTERMLINE,-LENGTH,0);
172 FERRMSG (FERROR,LTERMLINE,LENGTH);
173 PRINT (LTERMLINE,0,0);
174 PRINT (LTERMLINE,-LENGTH,0);
175 PRINTFILEINFO (FILE'NUMBER);
176 TERMINATE;
177 END;
178 $PAGE
179 BEGIN
180
```



Execution of the program begins on this page. The program banner and system time are displayed. The global RINs for the MONITOR program, program A, and program B are retrieved from the ;INFO= parameter with the GET'INFO procedure. The MONITOR global RIN is locked to restrict only one execution of this program.

Figure 2: Monitor program in SPL (Cont.)

```
181 <<.....>>
182 <<== Start of main program code. ==>>
183 <<.....>>
184
185 TRUE'COND:=TRUE;
186
187 MOVE TERMLINE:="(*** MONITOR *** B.01.00 Copyr. 1986, ",
188 "Senge Bruno. All rights reserved.)";
189 LENGTH:=TOS-LOGICAL(@TERMLINE);
190 PRINT (LTERMLINE,-LENGTH,0);
191 DATELINE (TERMLINE);
192 PRINT (LTERMLINE,-27,0);
193 PRINT (LTERMLINE,0,0);
194
195 GET'INFO (RUN'PARM, INFO'LENGTH, LINFO'STRING);
196
197 MONITOR'RIN:=BINARY (INFO'STRING,3);
198 IF <<> THEN
199 BEGIN
200 MOVE TERMLINE:="(*** Invalid monitor RIN passed. ***";
201 LENGTH:=TOS-LOGICAL(@TERMLINE);
202 PRINT (LTERMLINE,-LENGTH,0);
203 TERMINATE;
204 END;
205
206 PAUSE'RIN'A:=BINARY (INFO'STRING(3),3);
207 IF <<> THEN
208 BEGIN
209 MOVE TERMLINE:="(*** Invalid pause RIN for A passed. ***";
210 LENGTH:=TOS-LOGICAL(@TERMLINE);
211 PRINT (LTERMLINE,-LENGTH,0);
212 TERMINATE;
213 END;
214
215 PAUSE'RIN'B:=BINARY (INFO'STRING(6),3);
216 IF <<> THEN
217 BEGIN
218 MOVE TERMLINE:="(*** Invalid pause RIN for B passed. ***";
219 LENGTH:=TOS-LOGICAL(@TERMLINE);
220 PRINT (LTERMLINE,-LENGTH,0);
221 TERMINATE;
222 END;
223
224 $SPACE
225 <<.....>>
226 <<== Lock RIN 'MONITOR' to ensure that only one ==>>
227 <<== copy of this program is active. Locate the ==>>
228 <<== RIN number from the ;PARM= parameter. ==>>
229 <<.....>>
230
231 LOCK'COND.(15:1):=0;
232 MOVE BAL:=MONITOR ";
233
234 LOCKGLORIN (MONITOR'RIN,LOCK'COND,BAL);
235 IF <<> THEN
236 BEGIN
237 IF > THEN
238 BEGIN
239 MOVE TERMLINE:="(*** Monitor program is currently ",
240 "executing. ***";
241 LENGTH:=TOS-LOGICAL(@TERMLINE);
```



Now that the global RIN is locked, continue by opening the message files and enable the message file wait facility. Note the file and access options of the FOPEN intrinsic. This allows for multi-processing and inter-job access.

Figure 2: Monitor program in SPL (Cont.)

```

241 PRINT (LTERMLINE,-LENGTH,0);
242 END
243 ELSE
244 BEGIN
245 MOVE TERMLINE:="(*** LOCKGLORIN intrinsic error: RIN# = ",2;
246 NUM'CHAR:=ASCII (MONITOR'RIN,10,BAL);
247 MOVE ":-BAL,(NUM'CHAR),2;
248 MOVE ":-". RIN password = MONITOR. """,2;
249 LENGTH:=TOS-LOGICAL(@TERMLINE);
250 PRINT (LTERMLINE,-LENGTH,0);
251 END;
252 MOVE TERMLINE:"*** This process cannot continue. """,2;
253 LENGTH:=TOS-LOGICAL(@TERMLINE);
254 PRINT (LTERMLINE,0,0);
255 PRINT (LTERMLINE,-LENGTH,0);
256 TERMINATE;
257 END;
258 $PAGE
259 <<*****
260 <<*** Now open the MONITOR, PROGRAMS, and PROGRAMS ***>>
261 <<*** message files for read and write access. ***>>
262 <<*****
263
264 MOVE TERMLINE:"MSGM ";
265 MSGM'FILE:=FOPEN (TERMLINE, %30105, %2300);
266 IF << THEN
267 FS'ERROR (1,MSGM'FILE,1);
268
269 MOVE TERMLINE:"MSGA ";
270 MSGA'FILE:=FOPEN (TERMLINE, %30105, %2303);
271 IF << THEN
272 FS'ERROR (2,MSGA'FILE,1);
273
274 MOVE TERMLINE:"MSGB ";
275 MSGB'FILE:=FOPEN (TERMLINE, %30105, %2303);
276 IF << THEN
277 FS'ERROR (3,MSGB'FILE,1);
278
279 <<*****
280 <<*** Now enable extended waits on empty files for ***>>
281 <<*** read access and full files for write access. ***>>
282 <<*****
283
284 FCONTROL (MSGM'FILE, 45, TRUE'COND);
285 IF << THEN
286 FS'ERROR (1,MSGM'FILE,5);
287
288 FCONTROL (MSGA'FILE, 45, TRUE'COND);
289 IF << THEN
290 FS'ERROR (2,MSGA'FILE,5);
291
292 FCONTROL (MSGB'FILE, 45, TRUE'COND);
293 IF << THEN
294 FS'ERROR (3,MSGB'FILE,5);
295 $PAGE
296 <<*****
297 <<*** Now execute the main body of the loop waiting ***>>
298 <<*** for the control codes to process. ***>>
299 <<*****
300

```



The primary MSGM message file is read for incoming process requests. The external job streams for programs A and B may be started, stopped, suspended, or resumed.

Figure 2: Monitor program in SPL (Cont.)

```
301 OK:=TRUE;
302 TIME'2'QUIT:=FALSE;
303
304 DO
305 BEGIN
306 IN'LENGTH:=FREAD (MSGM'FILE, LINPUT'RECORD, -240);
307 IF <> THEN
308 FS'ERROR (1,MSGM'FILE,3);
309
310 IF IR'CONTROL'CODE < -4 LOR IR'CONTROL'CODE > -1 THEN
311 BEGIN
312 MOVE TERMLINE:="** Invalid control code of ",2;
313 NUM'CHAR:=ASCII (IR'CONTROL'CODE,10,BAL);
314 MOVE *:=BAL,(NUM'CHAR),2;
315 MOVE *:= " received. Ignored. **",2;
316 LENGTH:=TOS-LOGICAL(@TERMLINE);
317 PRINT (LTERMLINE,-LENGTH,0);
318 END
319 ELSE
320 BEGIN
321 I:=IR'CONTROL'CODE * -1;
322 CASE I OF
323 BEGIN
324 <<0>> ;
325
326 <<1>> BEGIN << ** Startup. ** >>
327 MOVE TERMLINE:=("STREAM PROGRAMA.STREAMS",%15);
328 COMMAND (TERMLINE,CERROR,CPARM);
329 IF <> OR CERROR > 0 OR CPARM > 0 THEN
330 BEGIN
331 MOVE TERMLINE:=("** Unable to stream PROGRAMA. ",
332 "Cerror="),2;
333 NUM'CHAR:=ASCII (CERROR,10,BAL);
334 MOVE *:=BAL,(NUM'CHAR),2;
335 MOVE *:= " Cparm",2;
336 NUM'CHAR:=ASCII (CPARM,10,BAL);
337 MOVE *:=BAL,(NUM'CHAR),2;
338 MOVE *:= " **",2;
339 LENGTH:=TOS-LOGICAL(@TERMLINE);
340 PRINT (LTERMLINE,-LENGTH,0);
341 OK:=FALSE;
342 END
343 ELSE
344 BEGIN
345 MOVE TERMLINE:=("STREAM PROGRAMB.STREAMS",%15);
346 COMMAND (TERMLINE,CERROR,CPARM);
347 IF <> OR CERROR > 0 OR CPARM > 0 THEN
348 BEGIN
349 MOVE TERMLINE:=("** Unable to stream PROGRAMA. ",
350 "Cerror="),2;
351 NUM'CHAR:=ASCII (CERROR,10,BAL);
352 MOVE *:=BAL,(NUM'CHAR),2;
353 MOVE *:= " Cparm",2;
354 NUM'CHAR:=ASCII (CPARM,10,BAL);
355 MOVE *:=BAL,(NUM'CHAR),2;
356 MOVE *:= " **",2;
357 LENGTH:=TOS-LOGICAL(@TERMLINE);
358 PRINT (LTERMLINE,-LENGTH,0);
359 OK:=FALSE;
360 END
```

In order to suspend each process, the secondary global RIN is locked by the monitor program. Once locked, program A or B is informed to unconditionally lock this RIN. This will be demonstrated later in program A.

Figure 2: Monitor program in SPL (Cont.)

```

361 ELSE
362 BEGIN
363 MOVE TERMLINE:={"Programs A and B have been ",
364 "started."},2;
365 LENGTH:=TOS-LOGICAL(@TERMLINE);
366 PRINT (LTERMLINE,-LENGTH,0);
367 END;
368 END;
369 END;
370
371 <<2>> BEGIN <<" Shutdown. ">>
372 LTERMLINE:=-1;
373 FWRITE (MSGC'FILE,LTERMLINE,-2,0);
374 IF << THEN
375 FS'ERROR (2,MSGC'FILE,4);
376
377 FWRITE (MSGB'FILE,LTERMLINE,-2,0);
378 IF << THEN
379 FS'ERROR (3,MSGB'FILE,4);
380
381 MOVE TERMLINE:={"Programs A and B have been stopped. ",2;
382 LENGTH:=TOS-LOGICAL(@TERMLINE);
383 PRINT (LTERMLINE,-LENGTH,0);
384
385 TIME'2'QUIT:=TRUE;
386 END;
387
388 <<3>> BEGIN <<" Suspend A or B. ">>
389 IF IR'PROGRAM'ID = "A" OR IR'PROGRAM'ID = "a" THEN
390 I:=1
391 ELSE
392 IF IR'PROGRAM'ID = "B" OR IR'PROGRAM'ID = "b" THEN
393 I:=2
394 ELSE
395 I:=0;
396
397 CASE I OF
398 BEGIN
399
400 <<0>> BEGIN
401 MOVE TERMLINE:={"Unknown process to suspend; must ",
402 "be 'A' or 'B'. ">>,2;
403 LENGTH:=TOS-LOGICAL(@TERMLINE);
404 PRINT (LTERMLINE,-LENGTH,0);
405 END;
406
407 <<1>> BEGIN
408 LOCK'COND.(15:1):=1;
409 MOVE BAL:= "PROGRAMA ";
410
411 LOCKGLORIN (PAUSE'RIN'A,LOCK'COND,BAL);
412 IF << THEN
413 BEGIN
414 MOVE TERMLINE:={"LOCKGLORIN intrinsic error: ",
415 "RIN# = "},2;
416 NUM'CHAR:=ASCII (PAUSE'RIN'A,10,BAL);
417 MOVE "":=BAL,(NUM'CHAR),2;
418 MOVE "":=" RIN password = PROGRAMA. ">>,2;
419 LENGTH:=TOS-LOGICAL(@TERMLINE);
420 PRINT (LTERMLINE,-LENGTH,0);

```



Program B is suspended by executing the code below. This completes the suspend code.

Figure 2: Monitor program in SPL (Cont.)

```

421 OK:=FALSE;
422 END;
423
424 LTERMLINE:=-2;
425 FWRITE (MSGA'FILE,LTERMLINE,-2,0);
426 IF <> THEN
427 FS'ERROR (2,MSGA'FILE,4);
428
429 MOVE TERMLINE:="Program A is suspended.",2;
430 LENGTH:=TOS-LOGICAL(@TERMLINE);
431 PRINT (LTERMLINE,-LENGTH,0);
432 END;
433
434 <<2>> BEGIN
435 LOCK'COND.(15:1):=1;
436 MOVE BAL:="PROGRAMB ";
437
438 LOCKGLORIM (PAUSE'RIM'B,LOCK'COND,BAL);
439 IF <> THEN
440 BEGIN
441 MOVE TERMLINE:="(*** LOCKGLORIM intrinsic error: ",
442 "RIM# = ").2;
443 NUM'CHAR:=ASCII (PAUSE'RIM'B,10,BAL);
444 MOVE ":=BAL,(NUM'CHAR),2;
445 MOVE ":=" RIM password = PROGRAMB. ***",2;
446 LENGTH:=TOS-LOGICAL(@TERMLINE);
447 PRINT (LTERMLINE,-LENGTH,0);
448 OK:=FALSE;
449 END;
450
451 LTERMLINE:=-2;
452 FWRITE (MSGB'FILE,LTERMLINE,-2,0);
453 IF <> THEN
454 FS'ERROR (2,MSGB'FILE,4);
455
456 MOVE TERMLINE:="Program B is suspended.",2;
457 LENGTH:=TOS-LOGICAL(@TERMLINE);
458 PRINT (LTERMLINE,-LENGTH,0);
459 END;
460
461 END. <<*** END OF CASE ***>>
462 END;
463
464 <<4>> BEGIN <<*** Resume A or B. ***>>
465 IF IR'PROGRAM'ID = 'A' OR IR'PROGRAM'ID = a THEN
466 I:=1
467 ELSE
468 IF IR'PROGRAM'ID = 'B' OR IR'PROGRAM'ID = b THEN
469 I:=2
470 ELSE
471 I:=0;
472
473 CASE I CF
474 BEGIN
475
476 <<0>> BEGIN
477 MOVE TERMLINE:="(*** Unknown process to resume: must
478 be 'A' or 'B'. ***)",2;
479 LENGTH:=TOS-LOGICAL(@TERMLINE);
480 PRINT (LTERMLINE,-LENGTH,0);

```



Programs A and B are resumed by simply unlocking the secondary global RIN. This will allow programs A and B to successfully lock the RIN. The RIN is then immediately unlocked and the process continues as before. Once the monitor program completes, the MONITOR global RIN is unlocked and the message files are closed.

Figure 2: Monitor program in SPL (Cont.)

```

481 END;
482
483 <<1>> BEGIN
484 UNLOCKGLORIN (PAUSE'RIN'A);
485
486 MOVE TERMLINE:="Program A has been resumed.",2;
487 LENGTH:=TOS-LOGICAL(@TERMLINE);
488 PRINT (LTERMLINE,-LENGTH,0);
489 END;
490
491 <<2>> BEGIN
492 UNLOCKGLORIN (PAUSE'RIN'B);
493
494 MOVE TERMLINE:="Program B has been resumed.",2;
495 LENGTH:=TOS-LOGICAL(@TERMLINE);
496 PRINT (LTERMLINE,-LENGTH,0);
497 END;
498
499 END; <<** END OF CASE **>>
500 END;
501
502 END; <<** END OF CASE **>>
503 END;
504 END
505 UNTIL (TIME'2'QUIT OR NOT OK);
506 $PAGE
507 <<*****>>
508 <<** Now close the files and unlock the global RIN. **>>
509 <<*****>>
510
511 UNLOCKGLORIN (INTEGER(RUN'PARM));
512
513 FCLOSE (MSGM'FILE,0,0);
514 IF << THEN
515 FS'ERROR (1,MSGM'FILE,2);
516
517 FCLOSE (MSGA'FILE,0,0);
518 IF << THEN
519 FS'ERROR (2,MSGA'FILE,2);
520
521 FCLOSE (MSGB'FILE,0,0);
522 IF << THEN
523 FS'ERROR (3,MSGB'FILE,2);
524 END;
525 $PAGE
526 END.

```

In Figure 3 below, Process A is diagrammed to use the ;INFO= parameter of the :RUN command to locate the program suffix of A or B. It also contains the primary and secondary global RINs for the password of PROGRAMx, where x is the program suffix.

The program issues a conditional lock on the primary global RIN. If successful, the program continues; otherwise, an error message is displayed since some other process currently has this RIN locked (this prevents two copies of the program from executing because another copy is already executing).

Program A opens two message files: the primary message file named 'A' is opened for read access; the monitor message file named 'M' is opened for write access in order to communicate with the monitor process M.

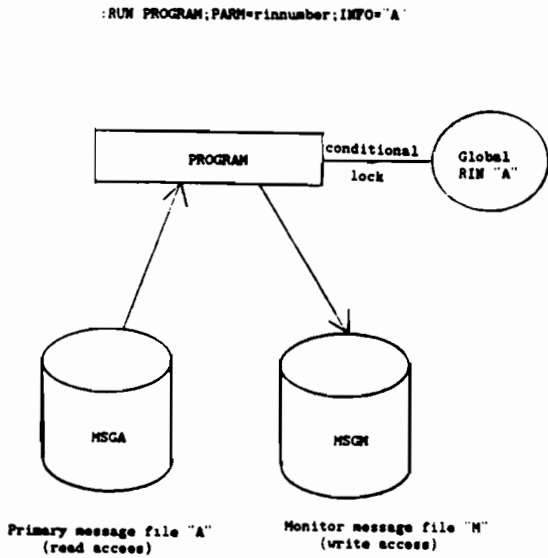


Figure 3: Program "A" Execution Flow Diagram

The second program example has been coded in COBOLII for comparison to that in SPL mentioned earlier. The variable names correspond closely to those used in the SPL version of the monitor program.

Figure 4: Program 'x' in COBOLII

```

1 $CONTROL SOURCE,MAP,CROSSREF,VERBS
1.1 IDENTIFICATION DIVISION.
1.2 PROGRAM-ID. PROGRAM.
1.3 AUTHOR. Benedict G. Bruno.
1.4 DATE-WRITTEN. June, 1986.
1.5 DATE-COMPILED.
1.6 REMARKS. The PROGRAM program will lock the global RIM name of
1.7 'PROGRAMx' with the RIM number passed in the ;INFO=
1.8 string and the 'x' replaced from the ;INFO=.
1.9
2 ENVIRONMENT DIVISION.
2.1 CONFIGURATION SECTION.
2.2
2.3 SOURCE-COMPUTER. HP3000.
2.4 OBJECT-COMPUTER. HP3000.
2.5
2.6 SPECIAL-NAMES. CONDITION-CODE IS COND-CODE.
2.7
2.8 DATA DIVISION.
2.9 $PAGE " "
3 WORKING-STORAGE SECTION.
3.1
3.2 01 TIME-2-QUIT PIC X(1) VALUE "N".
3.3 01 OK PIC X(1) VALUE "Y".
3.4
3.5 01 INPUT-REC.
3.6 03 CONTROL-CODE PIC S9(4) COMP.
3.7
3.8 01 DOUBLE-VARIABLES.
3.9 03 RUN-PARMD PIC S9(9) COMP.
4 03 RUN-PARMI REDEFINES RUN-PARMD.
4.1 05 RUN-PARMI1 PIC S9(4) COMP.
4.2 05 RUN-PARMI2 PIC S9(4) COMP.
4.3
4.4 01 LOGICAL-VARIABLES.
4.5 03 DOWE PIC S9(4) COMP.
4.6 03 LOCK-COND PIC S9(4) COMP.
4.7 03 TRUE-COWD PIC S9(4) COMP VALUE -1.
4.8
4.9 01 INTEGER-VARIABLES.
5 03 CERROR PIC S9(4) COMP.
5.1 03 CPARM PIC S9(4) COMP.
5.2 03 FERROR PIC S9(4) COMP.
5.3 03 I PIC S9(4) COMP.
5.4 03 IN-LENGTH PIC S9(4) COMP.
5.5 03 INFO-LENGTH PIC S9(4) COMP.
5.6 03 LEN PIC S9(4) COMP.
5.7 03 MSGM-FILE PIC S9(4) COMP.
5.8 03 MSGX-FILE PIC S9(4) COMP.
5.9 03 NUM-CHAR PIC S9(4) COMP.
6 01 PAUSE-RIM PIC S9(4) COMP.
6.1 03 PROCESS-RIM PIC S9(4) COMP.
6.2
6.3 01 CHARACTER-VARIABLES.
6.4 03 TERMLINE PIC X(79).
6.5 03 BAL PIC X(160).
6.6 03 CERROR-A PIC X(5).
6.7 03 CPARM-A PIC X(5).
6.8
6.9 01 INFO-STRING.

```



The procedure division follows below with the main driver section. The START-UP paragraph performs the initialization tasks, namely the retrieval of the primary and secondary global RINs as below.

Figure 4: Program 'x' in COBOLII (Cont.)

```
7 03 PROGRAM-NAME PIC X(1).
7.1 03 PROCESS-RIN-A PIC X(3).
7.2 03 PAUSE-RIN-A PIC X(3).
7.3 $PAGE
7.4 PROCEDURE DIVISION.
7.5
7.6
7.7 **** Start of main driver section. ****
7.8
7.9
8 MAIN SECTION.
8.1 PERFORM START-UP.
8.2 PERFORM PROCESS-INPUT UNTIL TIME-2-QUIT = "Y" OR OK = "N".
8.3 PERFORM FINISH-UP.
8.4 STOP RUN.
8.5 $PAGE
8.6
8.7 **** The initialization section displays the program banner, ****
8.8 **** locate the :INFO: parameters, and open files. ****
8.9
9
9.1 START-UP.
9.2
9.3 MOVE SPACES TO TERMLINE.
9.4 STRING "*** PROGRAM *** B.01.00 Copyr. 1986, "
9.5 DELIMITED BY SIZE
9.6 "Benge Bruno. All rights reserved."
9.7 DELIMITED BY SIZE INTO TERMLINE.
9.8 DISPLAY TERMLINE.
9.9 MOVE SPACES TO TERMLINE.
10 CALL INTRINSIC "DATELINE" USING TERMLINE.
10.1 DISPLAY TERMLINE.
10.2 MOVE SPACES TO TERMLINE.
10.3 DISPLAY TERMLINE.
10.4
10.5 CALL "GET'INFO" USING RUN-PARM'D INFO-LENGTH INFO-STRING.
10.6
10.7 IF PROGRAM-NAME NOT = "A" AND PROGRAM-NAME NOT = "a" AND
10.8 PROGRAM-NAME NOT = "B" AND PROGRAM-NAME NOT = "b" THEN
10.9 MOVE SPACES TO TERMLINE
10.9 MOVE "*** Invalid program suffix passed. ***" TO TERMLINE
10.9 DISPLAY TERMLINE
10.9 STOP RUN
11.1
11.2
11.3
11.4
11.5 IF PROCESS-RIN-A NOT NUMERIC THEN
11.6 MOVE SPACES TO TERMLINE
11.6 MOVE "*** Invalid process RIN value. ***" TO TERMLINE
11.6 DISPLAY TERMLINE
11.6 STOP RUN
11.7
11.8
11.9
12
12.1
12.2 IF PAUSE-RIN-A NOT NUMERIC THEN
12.2 MOVE SPACES TO TERMLINE
12.2 MOVE "*** Invalid pause RIN value. ***" TO TERMLINE
12.2 DISPLAY TERMLINE
12.2 STOP RUN
12.3
12.4
12.5
12.6
12.7
12.8 $PAGE
12.9
```

A conditional lock is executed for the primary global RIN as below. Once completed, the process and monitor message files are opened.

Figure 4: Program 'x' in COBOLII (Cont.)

```

13 **** Lock RIN 'PROGRAM' to ensure that only one copy of this ****
13.1 **** program is active. Locate the RIN number from the ****
13.2 **** :INFO= parameter. ****
13.3 ****
13.4 ****
13.5 MOVE SPACES TO BAL.
13.6 STRING "PROGRAM" DELIMITED BY SIZE
13.7 PROGRAM-NAME DELIMITED BY SIZE INTO BAL.
13.8 MOVE PROCESS-RIN-A TO PROCESS-RIN.
13.9 MOVE 0 TO LOCK-COMD.
14
14.1 CALL INTRINSIC "LOCKGLORIN" USING PROCESS-RIN LOCK-COMD BAL.
14.2
14.3 IF COMD-CODE > 0 THEN
14.4 MOVE SPACES TO TERMLINE
14.5 MOVE ** Program is currently executing. ** TO TERMLINE
14.6 DISPLAY TERMLINE
14.7 MOVE ** This process cannot continue. ** TO TERMLINE
14.8 DISPLAY TERMLINE
14.9 STOP RUN
15 ELSE
15.1 IF COMD-CODE = 0 THEN
15.2 MOVE SPACES TO TERMLINE
15.3 STRING "** LOCKGLORIN intrinsic error: RIN# = "
15.4 DELIMITED BY SIZE
15.5 PROCESS-RIN-A DELIMITED BY SIZE
15.6 " RIN password = PROGRAM. **"
15.7 DELIMITED BY SIZE INTO TERMLINE
15.8 DISPLAY TERMLINE
15.9 MOVE ** This process cannot continue. ** TO
16 TERMLINE
16.1 STOP RUN
16.2
16.3 $PAGE
16.4 ****
16.5 **** Now open the MONITOR and PROGRAMs message files for ****
16.6 **** read and write access. ****
16.7 ****
16.8
16.9 MOVE "MSGN " TO TERMLINE.
17 CALL INTRINSIC "FOPEN" USING TERMLINE %30105 %2303
17.1 GIVING MSGN-FILE.
17.2 IF COMD-CODE NOT = 0 THEN
17.3 CALL FS'ERROR' USING 1 MSGN-FILE 1
17.4
17.5
17.6 MOVE SPACES TO TERMLINE.
17.7 STRING "MSG" DELIMITED BY SIZE
17.8 PROGRAM-NAME DELIMITED BY SIZE INTO TERMLINE.
17.9 CALL INTRINSIC "FOPEN" USING TERMLINE %30105 %2300
18 GIVING MSGX-FILE.
18.1 IF COMD-CODE NOT = 0 THEN
18.2 CALL FS'ERROR' USING 2 MSGX-FILE 1
18.3
18.4
18.5 ****
18.6 **** Now enable extended waits on empty files for read ****
18.7 **** access and full files for write access. ****
18.8 ****
18.9

```



The extended wait facility is enabled as below. This completes the START-UP paragraph and the initialization function. The PROCESS-INPUT paragraph is called from the main driver to process input requests until a shutdown is received. The SUSPEND-PROCESS paragraph is executed when the monitor program requests us to suspend. This is accomplished by unconditionally locking the secondary global RIN as below.

Figure 4: Program 'x' in COBOLII (Cont.)

```
19 CALL INTRINSIC "FCONTROL" USING MSGN-FILE 45 TRUE-COND.
19.1 IF COMD-CODE NOT = 0 THEN
19.2 CALL "FS'ERROR" USING 1 MSGN-FILE 5
19.3
19.4
19.5 CALL INTRINSIC "FCONTROL" USING MSGX-FILE 45 TRUE-COND.
19.6 IF COMD-CODE NOT = 0 THEN
19.7 CALL "FS'ERROR" USING 2 MSGX-FILE 5
19.8
19.9 $PAGE
20 PROCESS-INPUT.
20.1
20.2 *****
20.3 **** Now executes the main body of the loop waiting for the ****
20.4 **** control codes to process. ****
20.5 *****
20.6
20.7 CALL INTRINSIC "FREAD" USING MSGE-FILE INPUT-REC -240
20.8 GIVING IN-LENGTH.
20.9 IF COMD-CODE NOT = 0 THEN
21 CALL "FS'ERROR" USING 1 MSGN-FILE 3
21.1
21.2
21.3 IF CONTROL-CODE < -2 OR CONTROL-CODE > -1 THEN
21.4 MOVE "*** Invalid control code received. Ignored. ***" TO
21.5 TERMLINE
21.6 DISPLAY TERMLINE
21.7 ELSE
21.8 IF CONTROL-CODE = -1 THEN
21.9 MOVE "Y" TO TIME-2-QUIT
22 ELSE
22.1 IF CONTROL-CODE = -2 THEN
22.2 PERFORM SUSPEND-PROCESS
22.3
22.4 $PAGE
22.5 SUSPEND-PROCESS.
22.6
22.7 *****
22.8 **** Now suspend the program by unconditionally locking the ****
22.9 **** pause RIN. When the RIN lock completes, we release ****
23 **** it immediately, thus resuming execution. ****
23.1 *****
23.2
23.3 MOVE 1 TO LOCK-COND.
23.4 MOVE PAUSE-RIN-A TO PAUSE-RIN.
23.5 MOVE SPACES TO BAL.
23.6 STRING "PROGRAM" DELIMITED BY SIZE
23.7 PROGRAM-NAME DELIMITED BY SIZE INTO BAL.
23.8
23.9 CALL INTRINSIC "LOCKGLORIN" USING PAUSE-RIN LOCK-COND BAL.
24 IF COMD-CODE < 0 THEN
24.1 MOVE SPACES TO TERMLINE
24.2 MOVE "*** LOCKGLORIN intrinsic error: RIN#*" TO TERMLINE
24.3 DISPLAY TERMLINE
24.4 MOVE "N" TO OK
24.5 ELSE
24.6 CALL INTRINSIC "UNLOCKGLORIN" USING PAUSE-RIN
24.7
24.8 $PAGE
24.9 FINISH-UP.
```



The FINISH-UP paragraph below completes program execution by unlocking the primary global RIN and closing the message files.

Figure 4: Program 'x' in COBOLII (Cont.)

```

25
25.1
25.2 **** Now close the files and unlock the global RIN. ****
25.3
25.4
25.5 CALL INTRINSIC "UNLOCKGLORIN" USING PROCESS-RIN.
25.6
25.7 CALL INTRINSIC "FCLOSE" USING MSGN-FILE 0 0.
25.8 IF COND-CODE NOT = 0 THEN
25.9 CALL "FS'ERROR" USING 1 MSGN-FILE 2.
26
26.1 CALL INTRINSIC "FCLOSE" USING MSGX-FILE 0 0.
26.2 IF COND-CODE NOT = 0 THEN
26.3 CALL "FS'ERROR" USING 2 MSGX-FILE 2.

```

In order to execute the monitor, A, and B programs, the global RINs must be located. The GETRIN MPE command is executed for each of the program passwords. Figure 5 below displays the logon and GETRIN commands. The bold face type denotes input user input.

Figure 5: Executing the GETRIN command

```

:HELLO BERGE.BRUNO RETURN
HP3000 / MPE V G.01.01 (BASE G.01.01). SUN, MAY 25, 1986, 10:10 AM
:GETRIN MONITOR RETURN
RIN: 122
:GETRIN PROGRAMA RETURN
RIN: 87
:GETRIN PROGRAMA RETURN
RIN: 139
:GETRIN PROGRAMB RETURN
RIN: 126
:GETRIN PROGRAMB RETURN
RIN: 129
:BYE RETURN

CPU=1. CONNECT=3. SUN, MAY 25, 1986, 10:13 AM

```

### The SHOWRIN program

The SHOWRIN program provides a formatted display of the system global RIN table. The program is written using privileged mode for execution on any MPE IV and MPE V based systems. The program may only execute in session mode and will immediately terminate in batch mode.

The group that SHOWRIN executes must have PM capability. For this reason, you may wish to place it in PUB.SYS or UTIL.SYS where PM is already in the group. The capability list of the user is checked when the SHOWRIN program is executed. If the user has system manager (SM) or privileged mode (PM) capability, then the SHOWRIN program may display the RIN table for all accounts; otherwise, only the user logon account may be used.

The SHOWRIN program may simply be executed using the :RUN statement. The alternate entry point of 'NOHELP' may be used to disable the initial help information as in Figure 6 below.

```

:HELLO BERGE,MANAGER.SYS (RETURN)
:RUN SHOWRIN (RETURN)

*** SHOWRIN *** 8.01.00 Copyr. 1986, Benge Bruno. All rights reserved.
SUN, MAY 25, 1986, 7:30 PM

The SHOWRIN program provides a display of global RINs on your HP 3000.
Each of the commands are entered as single characters as in OPT/3000.
The following commands are available.

A -> Prompt for a specific account name; search on this account only.
B -> Display both locked and unlocked RIN entries found.
E -> Exit the program.
H -> Display this help information.
L -> Display locked RIN entries only.
P -> Enable/disable printing to RINLIST on device LP.
R -> Prompt for RIN numbers for additional information.
S -> Display RINs for all accounts; Requires PM or SM capability.
U -> Display unlocked RIN entries only.
W -> Display waiting processes and the holding processes (deadlock?).

To disable this initial help dialogue, use the entry point of 'NOHELP'.

Please press any key to continue.

```

Figure 6: Running the SHOWRIN program



After pressing the return key, the system RIN table will be displayed with format in Figure 7 below.

System RIN Table		LP	System-wide, LOCK ONLY			6:05 PM
RIN	RIN	Creating				
#	password	Username.Acctname	PIN	Jobnum	Job name	
-----	-----	-----	---	-----	-----	

Figure 7: SHOWRIN header display

Note that the top line contains the current system time, LP, and RIN selection. Since the logged on user has PM or SM capability, the RIN table will be checked for all accounts. This is noted with the 'System-wide' literal. If PM or SM is not detected, then only the logged on account will be searched. This is noted with the 'Account: xxxxxxxx' literal.

Several dispositions of the global RINs may be selected. The 'LOCK ONLY' option displays only those RINs that are currently locked by a process. The 'UNLOCK ONLY' option displays only those RINs that are currently free. The 'LOCK w/WAIT' option displays only those RINs that are currently locked by a process and the processes that are waiting on these RINs. The 'LOCK, UNLOCK, WAIT' option displays all entries in the table.

When the print option is enabled, an asterisk (\*) is placed to the left of the LP designator in the header. The asterisk is removed when the LP option is disabled.

Entries found in the RIN table are displayed in ascending sorted order. For each entry found, the four digit RIN value, password, and creating user and account are displayed. If the RIN is currently locked, then the PIN, job or session number, and the job/session, user, and account of the locking process is displayed. An additional line is displayed with the PIN, job or session number, and the job/session, user, and account of any waiting processes.

The SHOWRIN program has single character terminal reads enabled every 20 seconds. Should no data be received, then a timeout occurs and the table is displayed using the same selection criteria.

To continue with our simple example of the monitor program and two external programs, we must first acquire the global RINs. This requires that the GETRIN MPE command be executed for the values of

'MONITOR', 'PROGRAMA', and 'PROGRAMB' with the program values executed twice; once for the primary process RIN and second for the secondary pause RIN.

Since the RIN values assigned by MPE are system dependent, I do not want to even suggest what they may be. However, using the SHOWRIN program, we may locate their values by using the 'U' for unlock or 'B' for all entries within the account which you have created these RINs. Having performed this in my own account, I may use the SHOWRIN program as in Figure 8 below to locate the (currently) unlocked RIN values and the associated passwords.

System RIN Table		LP	Account: BRUNO, LOCK,UNLOCK,WAIT			6:56 PM
RIN #	RIN password	Creating Username.Acctname	PIN	Jobnum	Job name	
87	PROGRAMA	BENGE.BRUNO				
122	MONITOR	BENGE.BRUNO				
126	PROGRAMB	BENGE.BRUNO				
129	PROGRAMB	BENGE.BRUNO				
139	PROGRAMA	BENGE.BRUNO				

Figure 8: Locating all RIN entries for a specific account

If the monitor, program A, and program B processes are initiated, then each program will lock its corresponding global RIN. Each process will await input by reading its associated message file. Figure 9 below shows this initial state, whereby each RIN value is displayed with the locking process.

System RIN Table		LP	Account: BRUNO, LOCK ONLY			7:11 PM
RIN #	RIN password	Creating Username.Acctname	PIN	Jobnum	Job name	
87	PROGRAMA	BENGE.BRUNO	153	#J397	A,BENGE.BRUNO	
122	MONITOR	BENGE.BRUNO	251	#S445	BENGE.BRUNO	
126	PROGRAMB	BENGE.BRUNO	121	#J398	B,BENGE.BRUNO	

Figure 9: Monitor, program A, program B normal execution

A useful feature of the global RINs discussed earlier is to use them for process control. If the monitor process were to lock the secondary pause RIN conditionally, programs A or B may be suspended if they attempt to lock the secondary pause RIN unconditionally. Figure 10 below displays this situation. You will note the previous entries as in Figure 9 for each of the processes primary RIN. The additional entries are showing that the secondary pause RINs are owned by the monitor process executed by session number 445. Programs A and B executing from job streams are then executing an unconditional lock for the pause RIN.


System RIN Table		LP	Account: BRUNO, LOCK ONLY			7:22 PM
RIN #	RIN password	Creating Username.Acctname	PIN	Jobnum	Job name	
87	PROGRAMA	BENGE.BRUNO	157	#J402	A,BENGE.BRUNO	
122	MONITOR	BENGE.BRUNO	159	#S445	BENGE.BRUNO	
126	PROGRAMB	BENGE.BRUNO	122	#J403	B,BENGE.BRUNO	
129	PROGRAMB	BENGE.BRUNO	159	#S445	BENGE.BRUNO	
139	PROGRAMA	BENGE.BRUNO	122	#J403	B,BENGE.BRUNO	
			159	#S445	BENGE.BRUNO	
			157	#J402	A,BENGE.BRUNO	

Figure 10: Suspending programs A and B

Figure 11 below displays only the pertinent information from Figure 10 above in that only those RINs with processes waiting are displayed. This is effective in reducing the entries in the display to detect potential deadlocks. Looking closely at the entries in Figure 11 below, we see that the primary RIN values are not included.

System RIN Table		LP	Account: BRUNO, LOCK w/WAIT			7:22 PM
RIN #	RIN password	Creating Username.Acctname	PIN	Jobnum	Job name	
129	PROGRAMB	BENGE.BRUNO	159	#S445	BENGE.BRUNO	
139	PROGRAMA	BENGE.BRUNO	122	#J403	B,BENGE.BRUNO	
			159	#S445	BENGE.BRUNO	
			157	#J402	A,BENGE.BRUNO	

Figure 11: Avoid deadlocks, display locked RINs w/waiting processes

	HP 3000	VS03/25
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

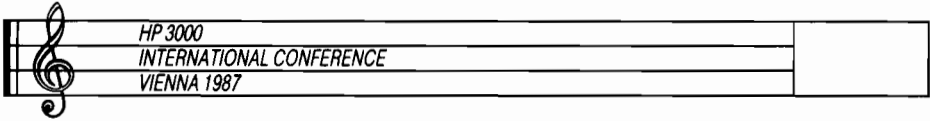
### Summary

Local and global RINs may be used effectively for the monitor and control of processes within an HP application. If RINs are used in conjunction with the MPE message file facility, complete process control of initiation, termination, and automated recovery techniques can be developed.

Global RINs can now be displayed quite easily with the SHOWRIN program. It is my hope that this program and/or a similar capability be incorporated into the HP Online Performance Tool (OPT/3000).

### Biography

Benedict G. Bruno has been working with the HP 3000 family of computer systems for seven years. His experience includes working for Hewlett-Packard Company as a Systems Engineer in the Los Angeles Airport Sales Office, a Network Consultant for Information Networks Division in Cupertino, and a Senior Applications Systems Engineer in the Rockville Sales Office. The integration of local and global RINs, message files, SPL, and HP data communications products for several application systems in retail, electronic mail, and others have required his need to develop the SHOWRIN program. He is currently president and cofounder of S.T.R. Software Company where he has developed POS/3000 to provide unattended data communications application software.



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

PETER R. ACKERMANN, Attorney-at-Law,  
ORBIT SOFTWARE International, London

THE LEGAL PROTECTION OF SOFTWARE IN EUROPEAN  
AND INTERNATIONAL LAW

1. Introduction

Ever since software had become a marketable commodity in its own right and consequently was copied by others for profit, the protection of software has been an important issue. The last decennium has seen new specific copyright protection laws in at least ten countries, including Japan, Germany and the US. With the growing development of more and more sophisticated software into virtually every country and its mounting importance vis-a-vis hardware, no national legal system can allow itself to abstain from studying the question whether and how computer software should be protected.

The UNESCO and the World Intellectual Property Organization (WIPO) have jointly considered all aspects of the protection of computer software at national and international levels in order to issue guidelines to national legislators who have yet to implement rules for this new field. WIPO has in fact in 1984 established the first technical classification of computer programs.

Most users of computer software have signed comprehensive licence contracts, containing non-disclosure clauses and regulating every aspect of their relation to the licensor. The drafting of software contracts is a lecture for itself. We are however not touching upon contractual relationships but are solely concerned with the situation between parties who have not signed one piece of paper.

## 2. Definition of "Computer Software"

What exactly is the object of legislation and jurisprudence, enacted and handed down for the purpose of our very protection? International consensus on a definition has not been reached but perhaps the most concise definition is given in the US Computer Software Copyright Act, which describes a computer program as

"a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result".

## 3. Mode of protection

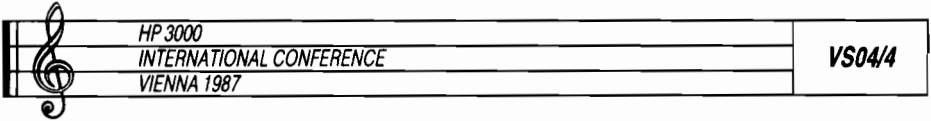
There are no less than eight different forms of protection that one can distinguish in the various countries that have already dealt with the subject matter:

- patents
- copyright
- trade secrets/confidentiality rules
- criminal law
- catalogue protection
- civil law
- protection against "slavish imitation"
- protection against "enrichment without cause"

3.1 At first sight, patents are often considered to be the natural way of protection for computer software, since patents and computers both are of a very technical nature. However, in most countries (e.g. West Germany, France, Italy and in the European Patent Act) the patentability of software is totally excluded. Several factors are responsible for this situation. Most countries limit patent protection to processes which are both novel and inventive. On one hand however, computer programs are not intended to solve problems in a novel way but computerize processes which more or less existed before. On the other hand, patent law is traditionally limited to processes of a physical nature and to physical products. Many computer programs concern methods of organization and administration which do not bring about any physical changes.

Under relatively severe restrictions, software can be patented in the USA. The restrictions, defined in a host of court cases, revolve around the underlying concept that a process may be patentable, natural laws are not. The discovery of a natural law or an algorithm does not convey a right to exclusive use thereof divorced of parti-





cular useful applications. It has repeatedly been held that "patent claims that seek to preempt the natural laws on which science and technology are built are not patentable since they inhibit rather than promote development of science and technology".

Japan, Holland and the UK also accept the patentability of software on the condition that it must serve a material process, or that an apparatus to which the program has been applied has become novel by it. Apparently, the possibility of industrial application is the key to a patent as may be the case with software-driven robots.

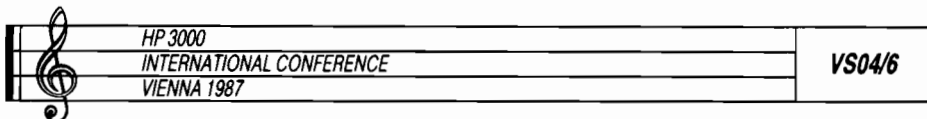
3.2 Copyright is the main form of protection for computer software. It is admitted in most countries. The US, the UK, West Germany and several others have already enacted specific computer copyright acts and legislation to that effect is pending in various others.

All countries require that works, in order to be protected by copyright, must meet a certain standard which is usually called "originality". Software must be an "individual creation of the mind", it must have a certain uniqueness, i.e. it must be expected that another programmer performing the same programming task would inevitably come to a different result. That of course is bland theory, because in any court proceeding, this "inevitability" will be hard to establish. Protection under Copyright is not only given for the program itself but also for the steps leading up to it. Flow charts are part of the copyright, underlying algorithms again are not. (Exception: in the UK, where algorithms seem to be protectable under copyright provided that they have been laid down in material form.)

Protectable is source-code and object-code, applications and operations software. Chips qualify for copyright protection in the UK, because the masks which are used are considered drawings, engravings or photographs. The US have enacted special legislation for the protection of chips which has a main feature of interest: If the chips are produced in a country that does not meet the reciprocity demands of the American legislation and are then imported, they do not enjoy protection under this particular act.

3.3 Most countries do not prescribe any formalities for copyright protection; exceptions are the US and Spain. In fact, the Berne Convention on Copyright does not permit the requiring of formalities with respect to foreign works. If we mark our products with the famous c in a circle, that consequently means that we think to enjoy a copyright, not that we have one. It will always be the courts to decide upon the originality of the program.

3.4 The rules with respect to ownership vary from country to country. In principle, copyright is vested in the author and can either not at all (Germany) or only in a limited way be transferred to another person. Consequently, only exploitation rights of software are exclusively or non-exclusively assigned to a third party. Almost all countries know a system by which copyright is vested in the employer if the program has been written by an employee or under contract.



3.5. The owner of copyright has the monopoly on producing, offering for sale and copying the software. Exceptions are found in the permissible copying for back-up purposes, laid down in law by the US and Australia and seen as "fair use" by other countries. Making single copies for private purposes may also be allowed if done for non-commercial reasons.

3.6 The adaptation of a computer program into another language or for use on another make of machine will be considered an infringement of copyright if made for commercial reasons. US law allows to adapt a purchased copy to one's own computer. If however, the modification goes further and the program is rebuilt with a modified structure, it may not be an infringement but a new copyrightable product. The reverse-engineered product will under these circumstances probably also not be considered infringing.

#### 4. Trade Secrets and Confidentiality

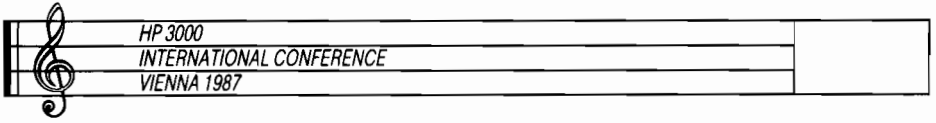
In principle, the law of trade secrets and confidentiality seems to be mainly available with respect to contractual parties, it cannot really be considered to be a form of third-party-protection. Trade secret law is not well suited to a mass market where confidentiality restraints are counterproductive to the author's endeavour to sell as many copies of his software as possible. Also, in an environment where employee mobility and entrepreneurship are common, trade secrets can hardly be protected.

## 5. "Computer Crime"

is the keyword which is much quoted but there are next to no specific measures in the general penal codes and more often than not we read, that as software "thief" was acquitted of theft, embezzlement or other "general crimes" which were defined long before the computer age.

Criminal sanctions of various forms have been introduced in most national Copyright acts. One can get up to three years in Germany for infringing copyright but I have never come across such sentences. Once caught red-handed, the civil law provides plenty of punishment because in general, the software owner will get damages amounting to the profits he would have made, had he sold all the copies that were made by the infringing party. That looks sometimes as if everyone would be better off if he waited for a strong marketer to nick his software and then reap the profits with the help of the courts later.

Computer software may or may not be protected for the original author. The fact that it says so as soon as it is installed may just be wishful thinking. As a ground rule, users should live on the proposition that the software they are finding on the market is protected; vendors should live on the proposition that it is not and that some technical means of protection is called for.



*HP 3000*

*INTERNATIONAL CONFERENCE*

*VIENNA 1987*

A PERFORMANCE COMPARISON OF HP3000 REPORT WRITERS

By Roger Lawson  
Proactive Systems Ltd  
Central Court  
Knoll Rise  
Orpington  
Kent BR6 0JA  
England  
(Tel:0689-77933)  
(Telex:9413362)

Running reporting programs or procedures often consumes a high proportion of the cpu and IO resources of an HP3000 computer system. This paper supplies data on how the commonly used report writers compare on performance. Also it takes a specific look at HPs new Business Report Writer (BRW) and sees how it stands up against other report writers (Hewlett Packard have been promoting the new BRW product as the answer to all reporting requirements on HP3000 systems). Having recently had the opportunity to try it out (and compare it to other reporting products performance wise) I have collected some interesting data.

Let me first declare that as I work for a company that produces an HP3000 writing product (namely Q-GEN), I will not attempt to present a full review of BRW or the other products - it would take more time than I have available to perform a full evaluation of each. However BRW particularly interested me because a couple of years ago I did an evaluation of QUERY, INFORM/REPORT (from HPs RAPID package), QUIZ (from COGNOS), ASK from COGELOG and a COBOL report writer (Q-GEN from PROACTIVE SYSTEMS). The evaluation was purely a comparison of their run time performance ie. how much load they placed on an HP3000 system when producing a typical sample of reports. To produce the data each sample report was written in each reporting language (not a simple task incidentally as it means you have to learn all the report writers and have a copy of the software on your system). The report procedures were then run against a medium size data base on a Model 44. The resulting figures were published in the HP IUG journal and presented in a couple of talks - a summary of the data is shown in Table 1 overleaf.

Table 1

Relative Performance in CPU seconds (COBOL=1)

	COBOL/ Q-GEN	QUERY	REPORT	INFORM	ASK	QUIZ
Test 1	1	2.1	1.8	2.1	1.9	2.3
Test 2	1	2.1	2.0	2.1	1.9	2.4
Test 3	1	4.7	8.7	7.9	1.9	2.2
Test 4	1	5.0	N/A	N/A	2.0	N/A
Test 5	1	4.4	N/A	N/A	2.0	2.0

Notes. N/A= Not Available. Relative performance in elapsed times were very similar and are therefore not shown. The tests range in complexity from very simple (test 1) to relatively complex (test 5) - see end of article for test details.

The figures highlighted what many people already knew. Namely that QUERY is a real machine killer, INFORM/REPORT are not much better, and that while QUIZ is better than QUERY it is still typically twice as slow as a reasonably well written COBOL program. Note that although claims have been made that INFORM/REPORT and QUIZ have been improved over the last couple of years since the original tests were done I doubt whether the figures would be significantly different today. Note also that the relative speed of the products does not vary much from one HP3000 model to another or between one configuration and another.

Now Hewlett Packard have been saying to some users that "BRW is not only very flexible but it can also be faster than COBOL" ie its the answer to the system managers prayers. To test this out I simply did some more comparative tests. I took two reports (one simple, one complex) and wrote them in QUERY plus rewrote them using BRW - I also converted the QUERY procedure to a COBOL program using Q-GEN. The three versions were then run on one of our mailing list data bases on our Model 37. The results are shown in Table 2 overleaf.

Table 2

Relative Performance in CPU seconds (COBOL=1)

	COBOL/ Q-GEN	QUERY	BRW
Test 6	1	2.2	2.4
Test 7	1	2.9	1.9


See end of article for test details. Relative performance on elapsed times were again very similar.

Well the first surprise is that on a very simple report, BRW is slower than QUERY! I didn't believe this when I first saw the figures so I reran the test with a similar result. Now BRW has one useful feature in that it gives you a breakdown on \$STDLIST of where it is spending the time when running a report - it appears that BRW is wasting a lot of needless time writing to and reading from a work file.

At least on a more complex report BRW seems to be a lot better than QUERY - it is probably more comparable in performance to QUIZ but it is still nowhere near a COBOL program (even one generated rather than hand coded).

Other comments - BRW has a very sophisticated, menu-driven report definition system with a compiler. I would have done more testing but the other users on our Model 37 complained that when I was running the report definition system, nobody else got a decent terminal response - as it takes a long time to step through the menus to produce a report this is not going to be a helpful feature on smaller machines (BRW seems to eat up a lot of memory). However if your other users can put up with it then BRW looks very powerful - but without doing the 4 day H/P training course I found it difficult to understand fully how to use the system from reading the reference manual. Even getting a copy of the manual from H/P proved difficult even though I was happy to pay for it - I had to borrow one from another user. BRW was written in Germany - it probably shows that in the number and comprehensive of the products facilities. One oddity was that I found I had to set up a dictionary for the test data base using DICTIONARY/3000. There is then a utility provided that is used to create an MPE file containing an extract of the dictionary - you can then throw away the dictionary (maybe H/P should provide short term rentals of DICTIONARY/3000 so that you don't need to buy it!).



	HP 3000	<b>VS05/4</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

If you are looking for a product as comprehensive as BRW then maybe the old RPG product is worth a look - it is certainly as flexible as BRW, is easier to learn, is industry standard and is well proven.

Note that BRW has many good points but end-user report writer it is definitely not. Non d.p. staff will even have difficulty understanding the HELP screens within BRW.

#### CONCLUSION


As an alternative to INFORM/REPORT for sophisticated users who wish to use a data dictionary (and don't mind the associated cost) then BRW warrants further evaluation. As an alternative to the free QUERY product (or other third party products) then BRW does not look so exciting. The effort to learn the new product (apart from the time to rewrite old QUERY procedures) is too high.

However QUERY is certainly slow - partly because of the interpreted nature of the product and partly because of the design of the record handling. Also it has certain limitations (such as lack of conditional printing and limits on the numbers of statements). To overcome these problems we developed a couple of years ago a compiler for the QUERY language which is called Q-GEN - it actually generates COBOL source code which it then links and compiles for you (or you can get the COBOL source and play about with it yourself). We also support extensions to the QUERY language such as an "IF" statement to do conditional processing, much higher statement limits etc.

This approach gives you much improved performance as you can see from the figures above - for example one of the original applications was to reduce an overnight reporting run from 15 hours (which meant the old version often did not finish until the next morning) to about 5 hours.

It also gives you total flexibility because if the QUERY extensions provided in Q-GEN are not enough you can always modify the COBOL code.

Even Hewlett Packard like the product - so far they are purchased about 30 copies for use in their own offices in different parts of the world.

	HP 3000	<b>VS05/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## TEST DETAILS

Tests 1 to 5 were run primarily on a Model 44 (with some repetition on a Series III). Tests 6 and 7 were run on a one megabyte Model 37 with two 7914 disc drives (running U-MIT and TurboIMAGE). All tests were run in job mode with no other jobs or users on the system.

Test 1 was a serial search of a detail dataset to retrieve 7000 records out of 59000. Report was an unsorted list of the records with no totalling and limited editing.

Test 2 was the same as Test 1 except that the records were sorted and only totals were printed (one level of sort).

Test 3 was the same as Test 2 except that a single data item from another data set was printed in the report total lines (item obtained from another detail data set via a common master).

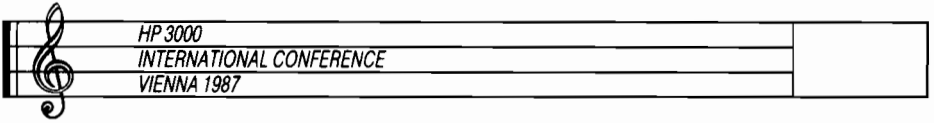
Test 4 was a serial search of a detail data set to retrieve 13600 records out of 92000 records. Report records were sorted and totalled at one level with an item retrieved from another data set as in Test 3.

Test 5 was a serial search from a detail data set to retrieve 6600 records out of 59000. Report records sorted and totalled and data from a linked master data set included in the report totals.

Test 6 was a serial read of 2731 records from a detail data set (out of 2732 records). The report was a simple list of two items from the data set with no sorting or totalling.

Test 7 was a serial retrieval from two detail data sets linked by a common automatic master (2700 records in each data set). 1870 records were selected for printing with sorting at 3 levels plus group totalling. Final report total, page numbering, heading lines etc also incorporated.

About the author: Roger Lawson is the managing director of Proactive Systems - an HP3000 software company based near London, England. He has a first degree in engineering and a masters degree in Business Administration. He has over 10 years experience of using HP3000s as analyst/programmer, dp manager, software supplier etc.



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

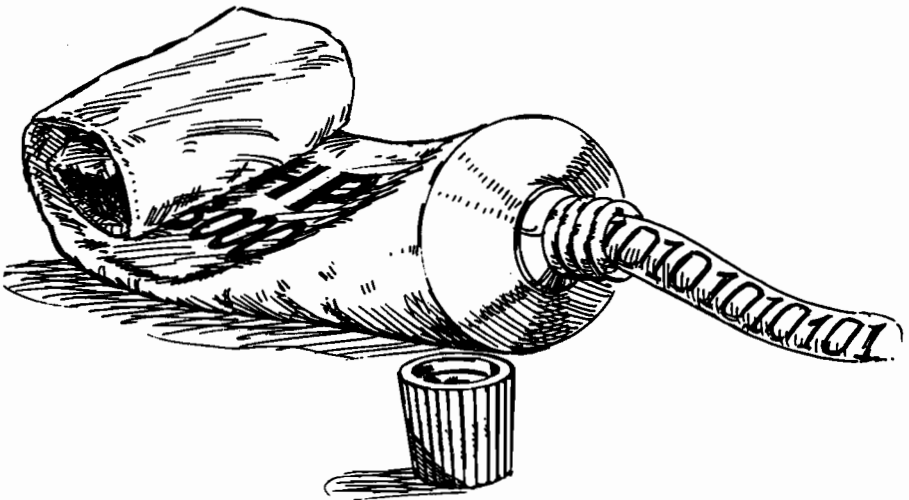
## What If ... You Didn't Wait For Spectrum?


-or-

## Squeezing the Last Bit Out Of Your HP 3000

By Michael Shumko and Robert Green

*Robelle Consulting Ltd.*  
8648 Armstrong Rd. R.R. No. 6  
Langley, B.C. Canada V3A 4P9  
Telephone: (604) 888-3666



	HP 3000	<b>VS06/2</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## What If ... You Didn't Wait For Spectrum?

-or-

### Squeezing the Last Bit Out Of Your HP 3000

By Michael Shumko and Robert Green

*Robelle Consulting Ltd.*  
8648 Armstrong Rd. R.R. No. 6  
Langley, B.C. Canada V3A 4P9  
Telephone: (604) 888-3666

**The problem.** Nightly batch jobs are still running the next morning. Users are complaining that on-line response is terrible. In short, your HP 3000 is over-worked, underpaid, and about to collapse from exhaustion.

**The solution.** Order a Spectrum: a Series 930 or 950.

**The problem with this solution.** Neither machine exists yet. (Okay. In the lab. But if it's not on the showroom floor, it can't be bought.) What to do?

**The solution to the solution.** We have done an informal survey of large HP shops to find out how the successful ones avoid topping out the HP 3000 line. What we found was not some "secret" formula, but rather a mundane, continuous, attention to the details of system performance. The successful sites still apply the long-touted answers for boosting performance, such as balancing use of disc drives. Just look in your magazines, newspapers and conference proceedings for all sorts of ways to improve performance.

Here are some of the ideas these users mentioned for how to 'squeeze the last bit from your HP 3000'.

1. use one cpu per problem (distributed processing).
2. distribute an integrated solution over several CPUs.
3. put heavy cpu work on PCs (word processing, graphics).
4. upgrade to faster hardware (Series 70, LAN, forms cache).
5. review overnight processing.
6. use NOBUF tools and optimum block sizes.
7. compile your fourth-generation applications.
8. get OMNIDEX for fast on-line database searching.

Not all of these solutions will apply to everyone. Many of these ideas are "old hat", but they work. A few of these ideas are novel - you may not have heard of them before. Some are not cheap (then again, neither is a Spectrum). If you're strapped for horsepower now, then these timely suggestions may give you the breathing room you need. Until Spectrum, of course.

## Use One CPU Per Problem

### Problem.

How do you add CPU power to a 3000 when you already have a Series 70?

### Solution.

Use one CPU per problem, or application, or department. Don't try to crowd everything onto one computer. Instead, use a separate CPU for each major application, or give each department its own machine. That way, you make each application independent of the problems in other applications. If the Payroll application is a hog, there's no reason for the Accounting users to suffer. Using separate machines also allows you to tune each machine for its own application. 'Distributed processing' was the strategy most frequently mentioned in our survey of successful sites. Most give programmers their own machine.

### Examples:

At Boeing, one of the large manufacturing systems has an 'update' machine and an 'inquiry' machine. The 'update' machine has 150 users who are updating the database. No uncontrolled inquiries or reports are allowed on this CPU. The 'inquiry' CPU has a copy of last night's database from the 'update' CPU; on this machine they allow people to make inquiries and to run QUIZ.


Longs Drugs, a large west-coast chain of drug stores, has 200 HP 3000s. An extreme example? Not really. True to the distributed processing ideal, each store has its own Series 37. These handle the main pharmacy application, keeping track of prescription stock, filling orders, and checking for dangerous drug interactions. When required, the Series 37s use dial-up DS to exchange information with the head office Series 70s. Otherwise, they're stand-alone machines. Every machine has a Console Engine to let the head office know when problems occur. (In fact, the Console Engine was initially developed for Longs Drugs.) At the head office, Longs puts separate applications on separate machines. For instance, all the Personnel applications are on one Series 70, the Accounting applications on another. Development is done on a separate machine again.

Consider another example, a company that sells supplies. They have 18 HP 3000s spread all over the world. Before the MIS manager went to work there, his philosophy was always 'get a bigger machine'. Then he went there, and they have a philosophy of 'getting the data down to the users'. So they have 3000s everywhere; every warehouse has its own small HP 3000. They were having a problem with FA/3000: they gave it its own Series 58. They don't even have a Series 70, and aren't budgeting for one until fiscal '88.

### Tools.

If you go this route you'll want to make sure that you have the proper tools to manage the network of machines properly. One type of tool is used to route spool files from one machine to another conveniently.

Unispool from Holland House is one example of this. This allows you to have an expensive peripheral like a laser printer connected to one machine, and have more than one computer send output to it.


	HP 3000	VS06/4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

**\$Stdlist Management software from NSD can also help; it checks spool files for error messages. This lets the computer look for problems itself, allowing the users to get on with their own work instead of baby-sitting the computer.**

If you setup a machine in a user department for unattended operation, you'll still have console messages to contend with. The Console Engine from Telamon attaches to the console and looks for specific conditions such as system failure messages, error conditions, and that sort of thing. If it sees that the system has run into some trouble it can either take action on its own (a 'pseudo operator') or it can dial the head office and notify the system manager.

**Resist getting a bigger machine.**

You can always have that in reserve if you get in trouble. Get another machine. The key advantage that users see, in addition to never 'topping out', is that you can push the machines into the user environment. You don't have to have a giant MIS. And when the machine is slow, it's because the users are running QUIZ reports. There are only a dozen users, so they can observe and figure it out, whereas on a Series 70 with 180 users, even the system manager doesn't know what's causing the problem. So you break it into smaller problems. Each machine is much less complicated, and I would guess, has much less problems. You will pay a little more for maintenance and raw horsepower, but you should be easily repaid in better user service.

	HP 3000	<b>VS06/5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Tip #2

### Distribute An Integrated Solution Over Several CPUs

Okay, I accept the idea that I should have one application per CPU, but my application is an integrated solution. All of the modules access common databases and I don't have time to rewrite it (or I bought the package and I don't have the source code).

#### Problem

You can't split a single integrated application over two machines.

#### Solution.

Yes you can, if you are clever.

#### AutoNet.

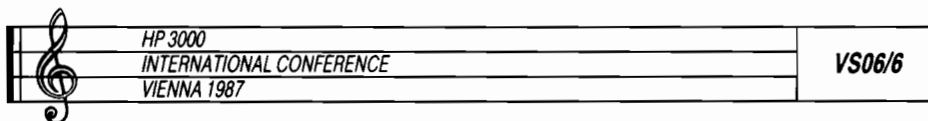
Karl Smith of Softsmith has developed an ingenious, simple method of distributing an integrated application over several HP 3000s. Compaq Computers started in business a few years ago. To manage their manufacturing work, they used ASK's MANMAN system over dialup lines to a time-shared Series 42. Within weeks they had their own machine, then two, and so on. Their growth has been so dramatic that they have never had the time to customize ASK's programs -- they use them "as is". Compaq now runs its entire company on a network of 900 PCs and seven HP 3000s (no IBM mainframe). When their processing needs for MANMAN exceeded the power of a single Series 70, Tom Callaghan hired Karl to program a solution.

Tom wanted to be able to spread the databases and files of the integrated MANMAN application over more than one HP 3000. Karl wrote an SL routine to intercept all calls to the FOPEN intrinsic. His routine, called Global FOPEN, checks the user's desired filename against a table of remote file-set names. If it doesn't find a match, Global FOPEN calls the real FOPEN. If it does find a table entry for the filename, Global FOPEN automatically gets the user a remote session with the same logon as his local session (unless he already has one), and calls FOPEN for the remote file. With this method, Compaq can easily distribute the ASK MANMAN package across several machines, with *no changes to the application*. Karl advises that there be a logical split in the application, where files may be moved. In the case of MANMAN, the three major components are purchasing, manufacturing and physical inventory. Users logon to the machine which contains the component they are interested in. This ensures that most of the database access is local, with only occasional access to files on the other systems. For more details, contact Karl! at (713) 332-3846 and ask about "AutoNet".

#### The Inside Details.

The software is not terribly tricky after all. The normal FOPEN is renamed to be HP'FOPEN, and their FOPEN routine is added to the system SL. When FOPEN is called, this routine determines which system the requested file resides on. If it's on another system, it just inserts the DS machine name into the device parameter, then calls HP'FOPEN. Nothing to it. If necessary, it opens a DSLINE and does a remote hello onto the other machine. In UB-Delta-1 the remote logon can be done automatically by NS as part of the DSLINE command, making Karl's routine even more vanilla. There will still be a remote CI process. All that is saved is the trouble of having to do the remote hello and remote bye. Another advantage is that this new feature takes one less NS socket.





## Tip #3

### Put Heavy CPU Work On PCs

Applications such as spreadsheets, graphics and word processing are notorious consumers of CPU time. These benefit from being on their own dedicated computers. PCs are a good choice, as a dedicated PC often performs better than a busy Series 70 on CPU-intensive applications.

Word processing is another application which definitely should be on a PC. If you're running HPWORD or some other word processing package on your HP 3000, you're paying dearly for it. You should not allow any word processing on your HP 3000 unless it's dedicated to word processing. HP has been advocating this approach for a few years, and the development thrust of their software has been in this direction, with more PC-based software, and access software to upload and download the data. You don't necessarily need the latest and greatest integrated software for uploading and downloading. Reflection from Walker Richer & Quinn will do the job fine.

#### Example:

Compaq Computers has some 900 PCs in their company. Instead of downloading raw data files from the HP 3000, they have summary files lying around which they download using Reflection, feeding them into Lotus or graphics or whatever. They do all their graphics on the PCs except for one giant run of 85 graphs in DSG, which comes at month-end on the laser printer. It ties up an entire Series 70 until it's over. They don't attempt to do anything else on that machine until the graphs are printed. But all of their other graphics, what-if graphics, presentation graphics, is done on the PCs. This keeps the graphics hogs off the HP 3000s.

## Upgrade to Faster Hardware

The Series 70 is a winner.

People we talk to say that their Series 70s are terrific, especially when they're loaded up with eight or nine megabytes of memory. They have a lot more horsepower than a Series 68. If you're on a Series 68 or smaller, you might consider going to a 70 instead of a 930 or 950. The Series 70 is so much more powerful than a 68 that we have heard that it is impacting the market for the Series 930.

When Longs Drugs upgraded one of their Series 68s to a Series 70, they went to U-MIT, Turbo IMAGE, and converted from Desk III to Desk IV all at the same time. At first they didn't see any difference in performance. But then they discovered that Desk IV ran 40% slower than Desk III! When they fell back to Desk III the system really took off! The extra power of the Series 70 masked the poor performance of Desk IV. Now that's horsepower, to be able to swallow up application problems as easily as that. When Boeing upgraded their TMS manufacturing machine from a 68 to a 70, they noticed a tremendous improvement in performance. Their 2-day backlog of batch jobs disappeared!


Use LAN/3000 instead of DS? A LAN will not reduce your system load, but users report that it offers much higher throughput than DS with just about the same overhead. You have to replace an INP with a LANIC, and string coaxial cable instead of regular wires. Bill Gates at Longs Drugs says that for the small price of 3% more cpu, a job which was taking 50 minutes over a 56 kb line using DS now takes eight minutes over the LAN. When Northern Telecom in Lachine went from DS to LAN, they got more communication throughput without noticeable increase of cpu overhead. They have three Series 70s, a Series 52, and two Series 9000s connected together in the same room. Besides being faster than DS it costs less, because they require only one LANIC per machine instead of many INPs. Using a 'vampire tap' they can add another machine to an active communications wire without affecting any other machines.

7933XP drives with hardware cache seldom help and can actually hurt performance. Perhaps the Eagle XP drives will work better. They have 2 megabytes of cache space, are 20% faster, and have reduced the "pep" overhead to one millisecond per access (from 6 or 10 ms.).

More memory can help, unless you already have 3 megs for caching.

New CRTs with Forms Cache (2394) can improve response time.

New 9600-Baud Modems from Microcom can make remote users smile. The Micocom AX/9624c modems understand HP's Enq/Ack protocol and have worked well on our Series 37 at Robelle. Remember, response time is perceived by the user, and a large part of that perception is not the processing efficiency of the programs, but the speed of the datacomm gear.

	HP 3000	VS06/8
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Review Overnight Processing

### Successful Sites Discourage On-line Reports.

If you allow users unlimited access to run reports on the production machine, then why should we feel sorry for you? You are getting the slow response that you asked for. Reports should run in batch, because that is where you can control the total number at any one time.

### The Night Time Is the Right Time.

To ensure good response for on-line users, most of the successful sites we contacted had a policy of strongly curtailing the number of concurrent batch jobs allowed during the day. The 3000 will run just fine all night long, without anyone watching it. Everyone we talked to was shifting work from prime shift to graveyard.

### What To Do When Overnight Jobs Don't Finish?

'Unfinished nightly jobs' is now a common complaint at HP shops, especially at month end (perhaps because people listened to advice to shift work to the evening hours). In our survey, we heard several methods for improving batch throughput: upgrade to a Series 70, get a separate CPU for reports, require department-head approval on job requests, reduce backup time, increase block sizes and, the most successful strategy, apply MR NOBUF tools wherever possible (as an HP SE said, "I have seen incredible speed improvement from front-ending QUIZ with SUPRTOOL. Software solutions to performance problems often show gains of 10 or 20 times. Hardware solutions, with no improvement in the efficiency of the underlying software, usually show gains of less than 1 or 2 times.")

### Backup Taking Too Long.

Many people are spending 2 to 4 hours per night on backup. If you run out of night, there are ways to reduce backup time. Get high-speed tape drives. Look at BackPack from Tymlabs. HP's Copycat program and the FCOPY-FAST option of MPEX will do a high-speed disc-to-disc backup, after which you can let the users and jobs on again and do disc-to-tape backup at your leisure. Elbert Silbaugh at Boeing uses this method and keeps his system available 23.5 hours a day. Another Boeing site in our survey wrote a privileged program to copy the database disc-to-disc *while the users are still accessing it in read-only mode*. Their system is available 24 hours a day. (Adager can also copy a database while it is open for read-only.)



## Tip #6

### Use MR NOBUF Tools and Optimum Block Sizes

**Problem.** One of the most common destroyers of system performance is the notorious **serial scan**. When you copy an enormous file, or reorganize a KSAM file, or select 100 records to report with QUIZ by reading every entry in a million-record dataset, you are bogging down the computer. The default methods of doing a serial scan are extremely inefficient on the HP 3000.

**Solution.** One of the most impressive ways to speed up serial I/O is to use MR NOBUF (multi-record non-buffered, not Mister Nobuff). You can write your own code to take advantage of MR NOBUF access if you're careful, but you don't need to - you can purchase tools that do it for you. Popular tools which use MR NOBUF access are HP's DSCOPY (you can use DSCOPY for copying files to the same system), HP's COPYCAT for file copying and backup, MPEX's FCOPY/FAST and Tymlabs' COPYRITE for file copying and duplication (powerful for KSAM users). Robelle's SUPRTOOL does MR NOBUF serial file access for IMAGE datasets (and any other file type) and Running-Mate replaces serial dataset reads in applications.

#### The Power of MR NOBUF.

We got a call a while ago from a fellow who didn't even know he had SUPRTOOL on his system, because it came bundled with another package he had bought. He found it, and the documentation, on his system so he started using it. He had a QUIZ job which normally took two hours to run, cruising through a huge database. A total novice, using the instructions in the manual he used SUPRTOOL to front-end his QUIZ report. The total time for this daily job went from two hours down to 15 minutes.

In actual tests, SUPRTOOL reduces the elapsed and cpu times to copy disc and tape files by 6 to 34 times, depending upon the blocking factor and record size of the file being copied. On a Series 37, FCOPY will copy 100 records per second (50 seconds for 5,000 records). A Series 68 boosts FCOPY to 400/second, but SUPRTOOL does 2,500/second, even on the Series 37 (25 times faster).

One of the shops we interviewed still uses a service bureau for some big accounting merges in IBM batch. They're considering that if the Spectrum is big enough, they might use it for that. They used to have four service bureaus. Now they're down to one. They brought things in-house by giving them their own machines, finding packages like mailing-list software front-ended by SUPRTOOL.

#### Block Sizes

The default blocking factors (number of records per physical disc block) is usually wrong. For big batch disc files, the maximum block size is now about 14K words (REC=14336), while the default is still the smallest block that will fit. The bigger the block, the faster the programs will run. For IMAGE databases the default block size is 512 words, as it has been since 1974. Many people we contacted in our survey were using 1024 words or more.

## Compile Your Fourth-Generation Applications

**Problem.** Interpreted Transact, and other 4GLs, consume too much CPU time.


**Solution.** Compile Transact source using the Fastran compiler.

When Cathy Vanderburgh was at Macmillan Blodel, she wrote up her experiences with Fastran as *Riding Herd on a CPU Hog*: "We recently developed a Transact system which included a large (15,000 lines) and complex (10 screens) data-entry program. After installation, the response times for the program varied from slow when the machine (an HP3000/64 with MPE IV) was lightly loaded to abysmal when the machine was heavily in use. Yet none of the other users on the system were experiencing similar problems at any time. We ran OPT/3000 to observe the execution of the program. The CPU time needed to interpret the IP code plus the complexity of the program was causing the MPE scheduler to class the process as a 'CPU hog' and to penalize it by dropping its execution priority. The only way to improve the response time would be to reduce the excessive CPU usage. Fortunately, this story has a happy ending. We discovered a piece of software called Fastran, a product of Performance Software Group, that compiles Transact source code into an executable program. On evaluation, we found that a Fastran version of the program used 1/4 to 1/3 of the CPU of the original Transact program, enough of a drop to bring the response back to an acceptable level. The user now enjoys(?) the same response patterns as everyone else on the machine. And the moral of the story? Without Fastran, of course, the author of the original program would now be busily re-writing it in COBOL. Plenty of programmers have discovered the hard way the functional limits of tools like Transact."

At CNR, where a large on-line application is written in Transact, compiling the application with Fastran led to a CPU reduction of over 60%, and a stack size reduction of 25%. Single-user elapsed run times did not improve much, but as more users were added, the reduced CPU requirements produced shorter elapsed run times. These numbers are for an I/O bound application where most of the time is spent in the database intrinsics and the file system; on CPU-intensive tasks the reduction can be considerably greater.

At Kitsap County they use Fastran over Transact wherever possible because the programs run much faster. However, they have found a few cases that Fastran cannot handle. Also, if a program needs extensive table handling, they choose COBOL over Transact.

Larry Kemp of HP Bellevue has found Fastran about 25% slower than COBOL and 50 to 98% faster than Transact (an 8 hour job reduced to 8 minutes was the best he ever saw!). An alternative 4GL that he found to give excellent performance is Protos; it generates a COBOL program for execution. And, finally, no one says you can't rewrite your most frequently used program in COBOL (use system logging to find out which program it is).

	HP 3000	VS06/11
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Tip #8

## Get OMNIDEX For Fast On-line Database Searching

IMAGE provides calculated read, chained read, and serial read. OMNIDEX adds record selection across multiple fields, generic retrieval and sorted sequential access, multiple keys in masters, and keyword retrieval on text data. It does this by adding another structure to IMAGE's: the binary tree. Traversing this tree is fast, fast, fast.

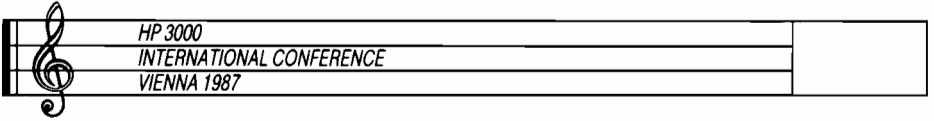
As you may have heard, HP uses OMNIDEX in the Response Center to index bug reports. That is how they can find out instantly who else has had a system failure 916 on Series 37 under T-MIT with a full moon. OMNIDEX indexes every word, not just the manually-assigned "keywords" as in the old SSB system. Doug Iles of HP says, "We could enter partial values and/or full values from several different fields and find 5 qualifying records out of 50,000 in seconds."

The people at D.I.S.C. (the suppliers of OMNIDEX) distinguish between "informational" data - data that you want on the system for doing inquiries, and "operational" data - data generated by the transactions of the organization. For example, in an order processing system, active orders are operational; customer and vendor master records are informational. Operational data is volatile and lightly indexed. Informational data is static and can afford to be highly indexed for fast, low-cost retrieval. In a general ledger system, the transaction dataset is operational. You do data entry and editing with it. When the transaction is completed, you post it to the ledger dataset, where it becomes informational data. You no longer modify it (much), but you need to ask numerous complex questions about it. OMNIDEX gives you the ability to index everything in your information data. You can use batch time to update the indexes, instead of on-line time.

Users also apply OMNIDEX to replace KSAM. The index-sequential part of OMNIDEX (called IMSAM) will reindex about 1 million keys per hour on a Series 70 (versus 20 to 30 hours with KSAM).

### Example:

Kitsap County Government is an HP site that gets a lot of work done without hitting the limits of the HP 3000 line. Jim Kellam, the manager, started with a Series 48, overloaded it, then added a Series 68 and left the 48 for development. He reports that OMNIDEX inquiries are unbelievably fast ('find all the voters named Smith' instantly replies '1200 entries found'), but can be abused, just like any tool. For example, one of their programs opens all eight databases at the start, in case you might need them. Installing OMNIDEX implies an extra open and another extra data segment, the equivalent of 16 DBOPENS per user. The users sometimes get in and out of the application to access other software, so they pay this startup overhead more than once per day. The IMSAM part of OMNIDEX allows you to define concatenated keys with pieces from 3 different datasets. Jim feels that they may have overused these features, because he observes slow response with some of these bizarre keys.



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987

**MIGRATING LARGE SOFTWARE SYSTEMS  
FROM AN HP-3000 TO A DEC-VAX**

Alexander Kotys  
SIS Datenverarbeitung Ges.m.b.H.  
Reithlegasse 4  
A-1190 Vienna, Austria

**Abstract**

Software portability is a major key in today's software development. Nevertheless almost every application is initially written to fit into a certain computer's architecture and environment.


Later on, if you think of transferring your acquired know-how in form of source code to another computer system you will find that it takes lots of efforts to adapt or implement all the nice and tricky features that made life easy and comfortable on your initial computer system.

This paper is intended to help you adapt software to be easily transferable to another adequate computer system and to reveal problems that might arise when moving existing well-running software to other computer systems.

I'll try to share my experience from moving a large business application, consisting of about 250.000 lines of Cobol source code and numerous utility procedures written in different languages, from an HP-3000 to a Digital Equipment - VAX computer system.

Finally I'll give you hints on developing standards and guidelines for writing software for 'generalized computers' keeping in mind that you actually have to implement this software on specific hardware and to integrate it with specific operating system utilities.



	HP 3000	VS07/2
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Introduction

This paper is intended to be a case study of a migration projekt at SIS where I was heavily engaged in migrating a large online payroll system (called LGS) from an HP3000 to a DEC-VAX computer system.

SIS is a Vienna based software house offering a broad range of standard software packages for the commercial market place. SIS has been working with HP computers for more than 10 years.

LGS stands for "Lohn- und Gehaltssystem" which is the german term for "payroll system". The whole application consists of an online menu program enabling the user to select from about 150 functions and a batch program that can execute most of those functions during off hours. More precisely, all functions not requiring human interaction after accepting input at the beginning can be processed in batch. Batch job launching is one of the interactive functions.

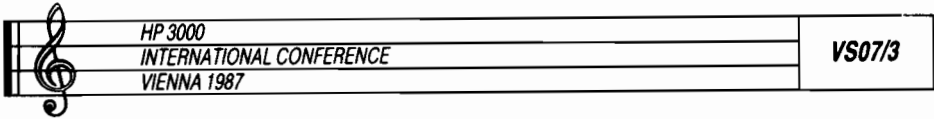
All our standard software packages, including LGS, make extensive use of HP standard software development tools, i.e. we exclusively use VPLUS for screen management, IMAGE for data management and COBOL'74 as programming language. Application programs do not directly call VPLUS or IMAGE procedures, but access them via standard interface routines written in SPL.

Auxiliary routines to perform tasks like date manipulation, interaction with the operating system, numeric conversions, text formatting etc. available to any program are written in SPL and PASCAL.

## The big decision

In the fall of 1985 management made the decision to broaden our customer base by offering LGS, that so far has been installed many times on HP3000 systems, on VAX computer systems. This decision was based on a feasibility study which aimed at identifying suitable "candidates" for a successful migration effort.

At that point it all began.



**The HP3000 is the one and only**

After intensive discussions we declared our HP3000 as the one and only source development machine. That means, we would maintain the sources of LGS only on the HP3000 and transfer them to the VAX for compilation purposes only.

This was one of the most important decisions and mainly based on the following considerations:

Being able to do source changes only once we could easily avoid developing different programs with certainly different behaviour and bugs.

Furthermore we would not need a separate development team, therefore reducing personal needs, keeping all know-how together and guaranteeing the same high level of technical support for our customers regardless of the computer system they are using.

Another consideration was to shorten the startup time by doing required source changes on the HP3000 using our well known development environment.

Later we found out that this was not quite realistic, because it did not save us from thoroughly learning the development tools and operating system features on VAX that we needed to write the necessary transfer and source conversion programs.

**Explanation of terms**

At this point I feel it's worth while to explain some terms and abbreviations for those of you who might not be familiar with Digital Equipment computer systems, in particular with the VAX family. Since probably all of you are quite familiar with the HP3000, I'll try to explain terms used on VAX by giving you the corresponding terms used on the HP3000 wherever possible.

The VAX family of minicomputers is essentially very similar to the HP3000 family, consisting of different models offering a very broad range of processor speed and computing power. The great difference is that all VAX processors implement a 32-bit architecture, i.e. a 32-bit bus structure as well as a 32-bit wide data and control path and registers. Similar is the extensive microcoded instruction set with numerous data types.

The standard operating system is called VAX/VMS that corresponds to HP's MPE in being an online-oriented multiprogramming operating system able to handle multiuser, realtime and multistream batch applications.


VMS, together with the VAX hardware, implements a virtual memory that is divided into pages of equal size. No segmentation exists, paging is done transparently to the user. Each page has assigned several attributes, such as a page type (code, data, etc.), access modes allowed for specific user classes and can be a process private, shared or system global page.

VMS offers extensive commands to perform the different tasks. These commands can further be tailored by so-called qualifiers that apply to the whole command or to selected parameters only depending on their position within the command. Every token can be abbreviated to any shortness as long as no ambiguity arises.

Other terms and abbreviations that will be referenced throughout this paper are RMS (Record Management System), the file system of VMS that handles sequential, index-sequential, relative and direct access files, terminal devices, mailboxes (corresponding to message files under MPE) as well as disk and tape devices just like the MPE file system does.

FMS (Forms Management System) is one of the several screen handlers available under VMS. It is somewhat similar to VPLUS although less features are offered. This tool will be presented in a separate chapter later on.

DBMS (Database Management System) is a CODASYL-compliant database

	HP 3000	VS07/5
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

system similar to IMAGE on the HP3000 in being a multilevel network database system.

RDB (Relational Database System) is, as its name states, a database system that provides the user with a relational view of his data.

DATATRIEVE is a query and maintenance language for data of all kinds. It is comparable to QUERY/3000 but has the advantage of being able to handle data stored in simple files, index-sequential files and different kinds of databases offered by DEC.

CDD (Common Data Dictionary) is a central repository for data about data. It describes the relations between data components. It is used by DBMS, RDB and DATATRIEVE.

#### Evaluating the COBOL compiler on VAX

Immediately after installation of the VAX system and learning the principles of VMS and its subsystems we started evaluating the COBOL compiler to detect any differences to the one available on the HP3000.

So we transferred some small-sized and some rather large sources to the VAX using an asynchronous terminal line with a kermit program on each end.

Since COBOL on VAX complies with ANSI'81 standards we did not expect too many problems. And in fact, most of them arose out of the use of HP extensions to the COBOL'74 standard or the use of machine-dependent constructs like calling intrinsics.

The differences found can easily be classified into several categories of which the most important are: differences in COBOL standard, HP extensions, DEC extensions and dependencies upon features of the HP3000.

Differences in COBOL standard are that the INITIAL clause of the PROGRAM-ID paragraph is required for the initialization of the

local variables each time the program is called (that is analogous to using the \$CONTROL DYNAMIC compiler option), and that the EXAMINE and the GOBACK statements are obsolete and have to be replaced by the INSPECT and the EXIT PROGRAM statements respectively.

HP extensions are such as the INTRINSIC phrase of the CALL statement, support of secondary entry points, DISPLAYing upon the operator's CONSOLE as a built-in function, the way to enter character constants in octal format and the possibility to specify compiler options within the source program.

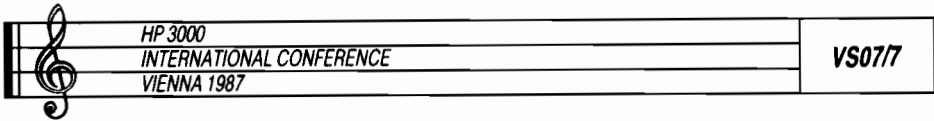
All those extensions are not defined by the ANSI-COBOL standard and therefore are not available on the VAX compiler. At least they are not available the way they are on the HP3000. In general we found that the VAX compiler is much more strict in enforcing proper syntax. Let's look at an example: suppose you have some copies to do some simple but tedious tasks, like a series of move statements or some standard calling sequence for a subroutine, and you code them free of periods, so you can include them right in the middle of an IF statement, then a sample use could be:

```

IF condition par ex. IF EVERYTHING-OK
 MOVE ...
 MOVE ...
 MOVE ...
 CALL "REPORT-SUCCESS" USING ...
ELSE ELSE
 MOVE ...
 MOVE ...
 MOVE ...
 MOVE ...
 MOVE ...
 CALL "REPORT-ERROR" USING ...
COPY modul-1.
COPY modul-2..

```

Please pay attention to the periods, the one behind "modul-1" just terminates the COPY statement, not the IF. The first period behind "modul-2" also terminates only the COPY statement, whereas the second period behind "modul-2" terminates the IF statement.



The same construct is possible on VAX too, but the standard says that after a period there must be a space, so the VAX compiler issues an error message indicating that there is no space between the two periods.

So on VAX the second COPY statement must look like


```
COPY modul-2. .
```

Strictness of the VAX compiler can be seen as laxity of the HP3000 compiler as well. In several instances the HP3000 compiler does not care too much about ANSI, it accepts paragraph and section names that begin in area B (after column 12), occurring data items defined as level 01 items and even accepts sections without paragraphs. This last problem arises in many programs and certainly is a left-over from the early times when we started programming in COBOL. We did not use sections in those days, our programs were made up of paragraphs only. Later we changed the paragraphs to sections to avoid the unreadable and error-prone PERFORM THRU construct, but forgot to insert paragraph names after the section headers. Since this does not bother the HP3000 COBOL compiler this has not been a problem until now.

Of the machine dependent features used within our COBOL programs only one is important enough to mention: the use of the condition code. It is returned by many operating system routines to indicate success or failure of the called routine. This would not bother us too much since we use interface procedures to most system routines, but unfortunately we ourselves wrote routines that return a completion status through the condition code.

Since there is no direct equivalent of the condition code on VAX we had to rewrite those procedures giving them one more parameter where they could return their completion status according to standard rules.

As DEC extensions to ANSI-COBOL I only want to mention support of statements to drive the DBMS database system, pointer and floating type data elements and an extensive ACCEPT/DISPLAY facility with full terminal enhancement and data conversion support. It does not

	HP 3000	<b>VS07/8</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

make sense to discuss this aspect any further, since those features are certainly not used by our programs and we will do our best to avoid the need of using them in the future.

### The automatic source code converter

Due to having several source constructs that cannot be equally fed into the COBOL compilers on the HP3000 and VAX respectively, we decided to provide some means of automatically converting sources coming from the HP3000 to meet the requirements of the COBOL compiler on VAX.


Contained within the procedure that transfers the source code is a program that changes the 80-bytes card images to variable text records, deletes line numbers, inserts paragraphs after section header where needed, commentizes the definition of the CONDITION CODE, replaces apostrophs ("'") by underscores ("\_") within external entrypoint names, adapts COPY statements to reference single files instead of members of a library, deletes \$CONTROL compiler options, inserts the INITIAL clause into the PROGRAM-ID paragraph and flags several other constructs that are known to cause problems but cannot be changed automatically.

Analogous to the source code conversion program we needed a tool to adapt our copylibs to our VAX-specific needs. This was easily accomplished by another program that puts each modul into a separate file within a special directory. We preferred this method rather than using a text library because we found that it was easier to handle within the source program.

### The screen handler

Things look great so far ! But with a bunch of compilable sources you still are far away from having a running software package !

Therefore we next looked for a replacement of VPLUS to handle all input from and output to the terminal screen.

	HP 3000	VS07/9
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

We started evaluating some of the terminal screen handlers available under VMS and tried to rate each one at internal structures, fulfillment of our needs, ease of customization to provide special features and last but not least, costs per installation. The final decision was in favour of FMS.

FMS was selected for several economical and technical reasons.

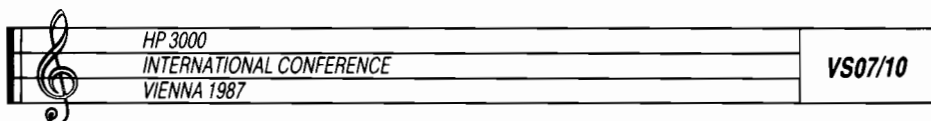
FMS is in its structure very similar to VPLUS. It offers an interactive forms editor and stores forms in a form library (equivalent to our forms file). Its runtime system consists of a very complete set of subroutines to perform the different tasks you would expect a good screen handler to do.

On the other hand some features of VPLUS are not covered by FMS, par ex. processing specifications and append forms. But both of these "missing capabilities" are no restrictions for us. Since introduction of family forms we changed all our head and append forms to families, and as to processing specifications, for a long time we wanted to move them into the COBOL code to speed up processing and to be able to react to input errors in a program dependent way. Now we took the chance and did it. Not only is the message "field can only contain digits" bad enough for a german-speaking country but even the german translation in general provides the operator with too little information. In several cases the ability to enter non-numeric data into a field depends on other parameters set by the application manager, therefore you have to inform the operator that right now he cannot enter non-numeric data while on other occasions he can do so.

#### Developing interface routines to FMS

After having made the decision to use FMS we had to write routines for our programs to access the FMS subsystem. Since we do not use the VPLUS intrinsics directly (there are only a few programs that make an exception to this rule) we had to implement only our interface routines instead of all VPLUS intrinsics along with the same set of interface routines calling them.





We checked all features of VPLUS we used and could classify them into the following groups: features directly mappable to FMS calls, features supportable through "own internal management" and a sequence of FMS calls, and features not supportable at all.

Obviously the easiest group to deal with was the one that contains the directly mappable functions like opening forms libraries, loading forms and displaying them on the screen along with the appropriate data.

Other tasks like getting input from the operator were not so easy to accomplish because VPLUS is strictly block mode oriented while FMS is not. Yet we needed a block mode-like behaviour since we did not want to change our programs to accept and process every single field. Therefore our interface routine that solicits operator input does quite a lot of work in simulating block mode. It has to prompt for each input field in screen order, has to check for primary field attributes (par ex. numeric-only fields, required fields), has to provide a means of moving the cursor around the screen and finally, has to perform data alignment operations depending on the field type. As to cursor movement, up to now we only support the left and right cursor keys, the forward and backward tab keys, a clear field key and the function keys, of course.

For the future we plan to support every cursor movement and special function key that exists on the keyboard.

Another interface routine that must perform what I call "internal management" is the one that emulates the VSETERROR intrinsic to indicate that data entered into a field is invalid in any sense and provide the operator with an appropriate error message, regardless whether the error has been detected by the input routine itself or later by the application program doing further input validation after form input has been completed.

Since it showed a negative impact on performance to reflect every change of data or enhancements on the terminal screen immediately we followed VPLUS implementing one bit per field to indicate whether this particular field is in error or not. Immediately

before new operator input is requested the screen is updated to reflect new enhancements as well as new data. Every field in error is then displayed using the global error enhancement, the other fields being displayed with their normal enhancement as specified in the form definition.

We chose the same strategy for optimizing the use of form families, a concept not known to FMS. Since all members of such a form family share the same layout it is not necessary to repaint the whole form on the screen when changing from one member of a family to another. All form specific information like field enhancements, field types and processing types (i.e. display-only or input field) is kept in an internal table provided by FMS but used by our routines only.

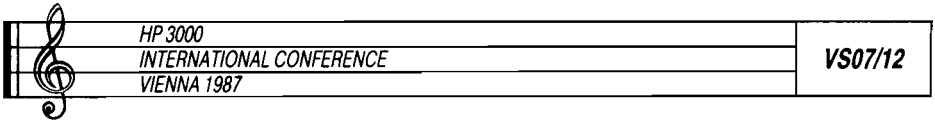
So the algorithm outlined above could easily be applied to forms belonging to the same family. From the point of FMS there is only one form on the screen whose fields change their enhancements every now and then. The decision whether to prompt for a certain field is not made by FMS but exclusively by our input routine which knows at each moment whether a field can accept input currently or is a display-only field.

#### Automatic conversion of VPLUS forms to FMS readable files

One of the nicest features of FMS, one I would like so much with VPLUS, is the ability to produce a simple readable text file out of a standard FMS form and vice versa, i.e. to translate such a text file describing a form in its details to the appropriate internal representation.

That was the way to transfer our VPLUS screens to the VAX !

We wrote a program on the HP3000 that displays each form on the screen, reads it from there, analyses field and text positions, gets various information from several VPLUS info calls and mixes all that stuff together to produce a file that FMS can translate into a form.



Another very nice feature of FMS that made it possible to provide our routines with all necessary information to emulate VPLUS is this internal table I mentioned above. It is called NAMED DATA.

For FMS NAMED DATA is just a bunch of strings of up to 80 characters that provide a form with application dependent information. This information is not bound to any object within that form. It can be accessed through two different keys, one being just an ordinal number while the other is an arbitrary name of up to 31 characters selected by the user.

What more do we need to access fields the way VPLUS does ?

The forms conversion program provides such a NAMED DATA for each field in the form. It stores field enhancement, field type, processing type and the VPLUS field name for documentation purposes.

When prompting for input the screen order of the fields must be known to jump right back and forth. This can easily be implemented using the ordinal number called field index that starts with 1 and consecutively increases by 1. Thus, jumping to next input field simply means reading the next field's NAMED DATA and checking if the field processing type stored within says that this actually is an input field. If this is not true the input routine has to continue with the next field until one is found that meets this criterion or no more fields exist in the form. In the latter case this means that input is finished and control should be returned to the calling program.

When a program wants to put data into a certain field or to retrieve data that already is in this field or to indicate that a field is in error it must provide the correct field number assigned by VPLUS. We implemented this access method by constructing a special name for each NAMED DATA consisting of a fixed string (like "F\$") followed by the VPLUS field number.

Every FMS call that references a field can identify it either by supplying the FMS field name or the field index that is equivalent to the screen order index. Taking as an example the emulation of a

VSETERROR call that supplies the VPLUS field number our corresponding routine would first construct the appropriate NAMED DATA's name out of the VPLUS field number, retrieve it and call the FMS procedure FDV\$AFVA (that stands for Alter Field Video Attributes) referencing the field through the field index being equal to the NAMED DATA's index.

### Restricted use of VPLUS features

As mentioned before, FMS does not cover all features of VPLUS we use in our forms files. Equally not every feature not supported directly by FMS can be emulated by our own interface routines.

Therefore we defined a list of VPLUS functions that we intended to support on VAX. This list has been held rather short and imposes restrictions on the use of VPLUS within our applications. On the other hand it will facilitate migration to other forms handlers since we expect to find almost all defined functions.


### The data management system

Selecting a data base system we proceeded very similar to the way we decided about the screen handler. But the result of this subproject was very different.

Again we first looked at databases offered for the VAX family. Almost everything exists to please your mind, from simple files over a hierarchically organized database up to the heights of a true relational view of data.

But all these systems have their drawbacks: they offer many more features than can be used by our applications and, most important, they are quite expensive.

Another reason for not using an existing database system was to stay as independent as possible from the concepts and strategies of hardware manufacturers and software suppliers.

	HP.3000	VS07/14
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

**Analysis of IMAGE features used by our application programs**

Migration of IMAGE applications does not require database systems with extensive features. IMAGE is one of the most stable database systems I have ever seen; but it is a rather simple database system, as well.

Analogous to VPLUS we access IMAGE databases through interface routines with integrated error checking. Eight such interface routines are enough to cover all operations possible within IMAGE.

Let me give you a very brief description of each interface routine.

The most important one is called RETRIEVE, it interfaces to DBGET and is used to read data entries from the database to the program's buffers. Seven different access methods are supported through DBGET (i.e. serial, chained, keyed, direct).

Chained RETRIEVES require a call to LOCATE (the equivalent of DBFIND) to set up chain pointers while serial RETRIEVES require a REWIND call to reposition the current record pointer back to the beginning of the dataset.

Data maintenance procedures are ADD, DELETE and MODIFY; their names are self-explanatory.

Concurrency is handled by the procedures LOCKSET and UNLOCK. As their names imply we restricted ourselves to dataset locks only, not making record or chain locks.

DBOPEN, DBCLOSE and DBINFO are the only IMAGE intrinsics directly called, but only the main program is allowed to call the first two of them and only two other programs need to call DBINFO.

**Development of NDM, a RMS based DBMS with the same interface as to IMAGE**

In developing a DBMS we decided to take the simple, straight forward approach. Our main goal was to provide all features of

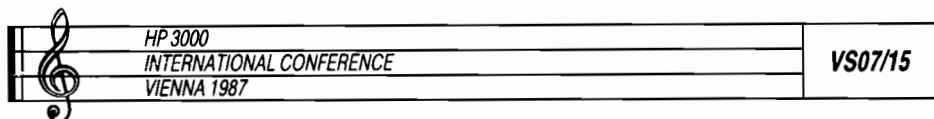


IMAGE we use rather than implementing all features IMAGE has. When support of a certain feature turned out to be very difficult to implement we rather decided to restrict its use within the application programs.

Chain length returned by LOCATE (i.e. DBFIND) was such a feature. Certainly we could have done it like IMAGE recording the number of corresponding detail entries along with the master entry, but we decided to dismiss it to avoid performance degradations.

To provide an IMAGE-like behaviour for our data files we decided to use index-sequential files as supported by RMS, the standard file system on VAX. It offered the best base with regard to functionality. Every dataset is put into a separate file and all the interface routines mentioned above are supported.


Since IMAGE is a two-level network database and our implementation is along the same lines we called it Network Data Management, NDM.

Clinging to our main decision to provide all interfaces as closely to the HP3000 as possible we wrote a program which interpreted exactly the same database schema file as used on the HP3000 to build all necessary file and data descriptions, data files and common data dictionary entries.

Keys for master datasets simply comprise the key item. No duplicates are allowed since they are forbidden in IMAGE, too.

Keys for detail datasets are constructed combining the respective search item with the sort item if one exists. We did not implement the extended sort feature of IMAGE because we observed very large keys. This was due to improper positioning of sort items within the dataset definitions resulting in a large number of items qualifying for extended sort. Rather we enhanced the schema syntax by adding an extended sort option to specify the last sort item in addition to specifying the first one as used by IMAGE.

This enhancement was done by adding special comment entries to allow these changes to be done on the HP3000. IMAGE just ignores

	HP 3000	VS07/16
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

these comment entries, but our schema processor interprets them and takes the appropriate action.

Serial RETRIEVES are implemented as serially reading along the primary key with the end of file condition being as usual. Chained RETRIEVES use the key specified by the last call to LOCATE and read serially along this particular key. The end of chain condition is raised whenever the end of file is reached or the search item part of the RMS key changes.

Considerable effort was necessary to compensate for the inability of RMS to serially read backward along a key. Therefore the schema processor has to define two keys for each chain, the first one as you would expect it, the other one to collect exactly in reverse order. To guarantee proper sorting sequence within duplicate keys we added a date-time field to each key in order to achieve unique keys.


Later on this double key technique was made selectable per chain due to performance impacts especially when adding many records to a dataset. Once again this selection was done through a special comment entry in the schema.

One of the next releases of RMS will support backward reading, thus, we will be able to completely eliminate these additional keys.

#### Restricted use of IMAGE features

I have just mentioned that we implemented only a subset of IMAGE, leaving out advanced features which are not needed by our applications. This led to a strict definition of the IMAGE subset allowed for the bulk of our application programs. Programs which perform very special tasks may rely on features not supported by our definition, but going beyond the defined limits requires these programs to be flagged as not being directly transferrable.

Some restrictions we imposed had no impact on our programs. They restricted only the use of features we did never use in our programs.

	HP 3000	VS07/17
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Since all our standard applications are self-contained and provide every necessary function from data entry to reporting there has never been the need to use data security at item level or multiple user categories with different passwords. User authorization is controlled by a master user of the application instead of a database administrator and security checking is done by the application programs themselves.

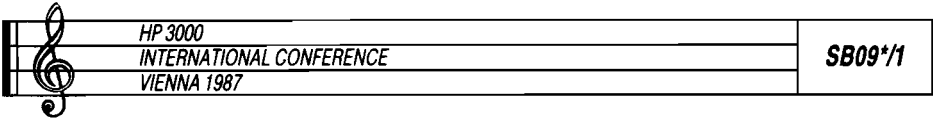
Also locking periods are held as short as possible, so we could restrict ourselves to lock whole datasets only.

Some restrictions caused source changes to our programs and, as stated at the beginning, we performed them on the HP3000 in order to maintain only one source version. This will certainly be the method of choice for all future changes as well as for migration to other computer systems.





HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987



**THE OPERATORLESS HP3000  
(Fact or Fiction?)**

**HP3000 International Conference  
Vienna 1987**



Henk Alblas  
Hewlett-Packard (Canada) Ltd  
6877 Goreway Drive  
Mississauga, Ontario, L4V 1M8

## INTRODUCTION

THE OPERATORLESS HP3000 - "FACT OR FICTION"?. The answer to this question is probably yes meaning that in certain scenarios, it is a fact and in other scenarios it is not possible to run the HP 3000 operatorless. Also, it depends on the definition of an Operatorless System, as certain tasks that can not be automated can be qualified as operator or end user tasks. These tasks are typically peripheral operations such as loading paper in printers or mounting cartridge tapes or reel-to-reel tapes on tape drives. However, the objective in operations automation is not necessarily to eliminate operator tasks completely. It can also be aimed at eliminating the need for a human operator at night, minimizing the amount of time required for operations tasks or minimizing the operator expertise level such that no special personnel is required.

The HP 3000 family members have always been recognized as systems that are easy to operate. This means that the amount of operator attention required for an HP 3000 is typically far less than of some competitive systems. But HP 3000's have grown in the number of tasks that they perform and in many sites, multiple systems are installed. These facts combined with the fact that labor costs for computer professionals are very high has led to the need for more organizations to look at operations automation.

Costs are indeed the main reason why we look into the process of making computer systems operatorless. Reduced operations staff, of course translates directly into cost savings. However, there are other benefits to operations automation. If done right, increased up time, higher system utilization and a more consistent environment will also result from this process. These factors are also cost savings, but are more difficult to quantify in monetary terms.

There are of course other scenarios where an operatorless environment is a must. If the HP 3000 is installed in an end user department or as part of a network in branch offices or retail stores of a large organization, local operator expertise will normally not be available. Here, unattended operations or at least operations whereby the expertise level of the operator is reduced, are required. Typically, in such cases there is a support group implemented at a central site to monitor and support the systems.



What are the components of operations automation? To achieve operations automation there are four areas we have to address. Firstly, batch processing has to be looked at to ensure that batch activities, both in the daytime as well as night, are performed and that if error conditions occur, proper recovery procedures are taking place. The second component is system backup. To ensure that the system is protected against fatal errors, system backups will have to be performed on a regular basis. This ties in with the third component which is system integrity. Maintaining the system's integrity for a system where there is no operator or reduced operator attention requires some additional measures. The final component of operation automation is remote operations and maintenance. Realizing that no environment is completely static and knowing that not all error conditions can be anticipated, it is required that corrective action can be taken. Normally this will take place in a "remote" fashion.

In the rest of this presentation we will be looking at these four components and I'll be trying to identify the issues, provide you with generic solutions to these issues both utilizing standard facilities and commands, as well as identifying functionality provided by third party software and hardware vendors. The intent of this paper is that it will provide you with a starting point to build upon in the process of automating your operations.

#### **NETWORKED AND STANDALONE SYSTEMS**


Before we look at the individual components of operations automation, we have to realize that there are differences between stand alone environments and networked environments. In a network environment, typically a central node, or head office system does provide for coordination of the network and ensures that the remote systems are monitored. This also means that the central node location is typically staffed with human operators. The remote locations in these networks are very similar and the configurations identical. The central node may be permanently linked to the remote locations or communications may be in place through dial-up links. In a smaller network the central node may still be in place but in this case, this node may not be staffed. Here the central node can still monitor the remote locations, however if the central node is interrupted, the operations of the remote systems may also be jeopardized. In this case an alert should go out to technical support. In a stand alone system environment it is also necessary that if an irrecoverable error occurs an alert should go out to the appropriate technical support person, this being the system manager or an on duty person. I'll come back to this remote alert scenario when I talk about remote operations and maintenance.

## BATCH PROCESSING

Let's look now at the automation of the batch processing activities. In most installations, batch jobs are run throughout the daytime in limited quantities and the major batch processes take place overnight. The decisions that have to be made in a batch processing environment are when to run the batch job, under what conditions, and if an error occurs what corrective measures to take. MPE facilities to aid in this process are the STREAM command, special JCW's, and conditional statements available to job control. The STREAM command allows for streaming of a job based on date and time or it can be scheduled in a certain amount of time from the moment streaming takes place. Even day of the week or days from the end of the month are options that can be used to determine the proper run date and time for the batch job period. This in combination with special JCW's that provide current date, time and day of the week allow for a very flexible scheduling mechanism. The conditional IF statement allows for decision making within the job stream. The IF can test JCW's, including JCW's set by user programs or HP subsystems; based on the results of this test different paths through the job stream are taken. Job streams can schedule other job streams to continue processing or to take corrective action. The job stream can even re-stream itself for a time that allows for the corrective procedure to complete. Examples of some of these job stream facilities can be found in the communicator for T-Mit and later versions of MPE.

Not many job streams utilize what I would call typical online commands. These commands, such as LISTF, can be used to determine the flow of the job. The way the LISTF command would be used in this case would be to issue a LISTF for a specific file and subsequently to test the JCW CIERROR for its value. This JCW will indicate if the file was found (CIERROR 0) or that the file was not found, CIERROR 907. Of course this is only a simple test and would not indicate to the job stream if the file is currently opened by another process. However it shows how some of these commands can be utilized.

In using standard MPE commands and utilities, HP 3000 users have found limitations. One of these is the inability to intercept jobs streamed directly by the user or an application package or program. Potentially this job could collide with other scheduled job activities. Other limitations are the lack of flexibility in a dynamic environment to check all the parameters and also the fact that it is difficult to allow job streams to issue console commands such as ABORTJOB and LIMIT.

	HP 3000	<b>SB09*5</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Because of the above mentioned limitations, third party job management or job scheduling packages that address these are quite attractive . Also these products provide features not mentioned here in most cases as well as an easy to use interface to set up the operating environment. Make sure to include in your evaluation of vendors, attention to their ability to provide ongoing support. This is especially important where privileged mode facilities are used, which is often the case with the products referred to here.

When something goes wrong the normal course of action should be an automated recovery procedure. This assumes that the point of failure and the reason for it is predetermined or at least that a catch all test was satisfied. If the failure and corrective action cannot be anticipated or recovery requires manual operations such as mounting backup tapes, an automated recovery may not be possible. At that point it is important that, depending on the urgency of this job stream, that the process be aborted or an alert sent out. Again, I'll come back to the alert in the remote operations and maintenance section.

As mentioned before, the job scheduling environment should cover both day and night time. In the day time, typically, some jobs will be executed. Also it could be made part of the batch processing activity to initiate on line user access. The MPE STARTSESS command can be used for this purpose. This would eliminate the user having to log on to the system. Of course security measures have to be taken, for example the first screen of the application should ask for the password. An example of an application that could be used to provide this as well as allowing for applications to run from that environment is HPDeskmanager. Note that version IV of HPDeskmanager is required to run applications from within this product.

At the end of the day again a batch process could take responsibility for shutting down the applications. Of course, procedures should be in place to allow users sufficient time and warning before the actual shutdown occurs. This automatic startup and shutdown of activities will also increase the consistency towards the end user of the availability of these applications.


Unattended printing may be a problem. Printers are mechanical devices and printer jams are not a thing of the past yet. In addition to this printer requirements as far as non-standard stationery goes also have to be met. This means that even if overnight printing takes place it is important to ensure that sufficient disk space is available for all spooled outputs that could be generated overnight. This in case that the printer would jam or be unavailable throughout the night. Printing of documents that require special forms should be done in the day time. All this assumes that the paperless office is not a reality yet. If your office is typical, you know that today more paper is generated than just five years ago.

## SYSTEM BACKUP

The next component of operations automation I want to talk about is the system backup. We all realize that the value of the information stored on our computer system in the form of data is very large. To protect this data against loss in the case of an emergency, frequent system backups are required. A system backup strategy has to be chosen that allows for recovery of data up to a point where transactions can be re-entered or where the loss of those transactions is allowable. A typical backup strategy is a weekly full backup plus partial or incremental backups daily. I won't go into a discussion of the tradeoffs between the frequency and size of system backups versus the impact on operations. However from an operations automation standpoint, it is beneficial to try to minimize the size of the system backups. The backup volume is important because of the fact that the different backup media have only a limited capacity per volume. If this limit is passed a more expensive solution is required or even manual operator intervention is needed. The different backup media are magnetic tape, cartridge tape, cartridge autochanger, and disk using HPCopycat. The amount of information stored on the different media varies: magnetic tape - approximately 30-40 megabytes per reel at 1600 BPI and approximately 130-140 megabytes per reel at 6250 BPI; cartridge tape drive with up to 67 megabytes, cartridge autochanger which will store up to 536 megabytes and the HPCopycat scenario where the backup can take place to any number of disk drives, allowing for even the largest volume; however this will also be the most expensive solution. As long as the tape based backups do not take more than one volume, the backups can be automated through the autoreply feature of the system configuration.

Note that third party packages are available that provide data compression on the backup process. This will quite often save significant amounts of space on your system backup medium and potentially allow you to limit this to one volume.

An alternative scenario is to utilize HPSilhouette to mirror data base information on the same system or on an additional system such that the mirror data base can be backed up while the primary data base is in use by the end users. This allows for system back ups in the daytime when the operations staff is available. Again this would apply in the case that a multi-reel backup is required. Note that third party software is available to allow this function to take place for files other than image and Turbolimage databases.

	HP 3000	<b>SB09*7</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## SYSTEM INTEGRITY

To maintain system integrity in a system that is running operatorless requires additional activities and safeguards. If the environment is static a database can be set up to indicate what files should exist on the system on a permanent basis. This database can be referenced to eliminate any files that are not in the database. To do this a custom program is required. This program can make use of a new intrinsic, FLABELINFO. This intrinsic provides information from the file label. This can be used to ensure that not only the file exists, but also that the file characteristics are matching the information in the database. After this check has been completed the system free space has to match a predetermined value, assuming that no disc free space has been lost. Care should be taken to maintain the database in sync with changes to the application software as well as the system software. Note that this also includes a new version of MPE, because this will typically mean different file sizes, additional files or changes to file names. Changes to system parameters such as the size of virtual memory will also impact this information. In some environments, especially large networks, it may be beneficial to validate that the set of programs installed on the remote systems match a predefined pattern. This can be done by keeping a small database, either locally or on the main node in the system. In addition to the checks mentioned before, filenames and file characteristics, it is possible to run a hashing program on the program file. This will show if any patches have been applied to the program, as this would not be reflected in the file label.

Security requirements of operatorless system are higher than other environments. We can divide these security issues in two groups, being security measures against access from outside versus measures to insure local security. Access by remote users, including remote console users, can be protected in different ways. This to prevent intruders from easy access to your system and your valuable data. If access for remote users has to be made available there are different solutions in the form of additional security software packages or additional hardware solutions, for example dial back modems. Furthermore the ports used for remote access can be "downed", through the MPE DOWN command outside of normal hours. Protection of the console or remote console port may be more difficult if these ports are to be used for Technical support personnel to check on the status of the system or perform remote operator tasks. Again addition levels of security software or hardware solution can be used to protect these ports.

A third party hardware solution is available to achieve the protection of your console. This product requires a hardware unit both at the system location as well as the remote operator's location. In addition to providing a more secure environment this solution also automates remote alerts, both intercepting console messages and acting upon a down system situation.



Local security is very similar to security needed for a system operated by human operators. This means that physical security should be in place to disallow unauthorized access to the system hardware components. For activities that have to be performed by end users, such as loading paper in a printer or mounting backup tapes, the particular peripheral devices can be placed outside of the secured area. Of course backup tapes mean a security risk. A system backup tape can be used to extract all system passwords. Naturally this jeopardizes security. Therefore these tapes should be kept in a vault or other suitable area. Off site copies should be taken on a regular basis and these should be treated in the same way.

In addition to these issues, which may be specific to an operatorless environment, standard security rules should be adhered to; assigning passwords to all accounts, changing these regularly, set up access rights tightly (only access to what is needed) fall in this category.


#### **REMOTE MAINTENANCE AND OPERATIONS**

An issue that comes up in automating the operations of a HP3000 is the need for a real console. Also the fact that taking the console across a DS link is not supported limits the options in certain scenarios. A new HP product HPEasytime that will initially be only available for the Micro 3000 provides a easy to use operator interface. It also eliminates the need for a real console. This product will be available for all HP3000's in the future.

To address the need for a remote console over DS (or NS) links, HP has a product by the name of INCS/3000. As this product is not generally released; contact your HP representative or HP Networking Consultant for local availability and details.

Referred to earlier in my presentation was third party hardware that works closely with the console to provide security for this essential access to your system and secondly this product monitors console activity. This means that it is able to intercept TELLOP messages sent by job streams and programs and act upon these with an autodial to a central monitoring site and deposit an alert message. This product will also detect system interrupts, system failures, system halts and hangs.

Remote operations can be performed from a central location in the case of a large network or from the technical support personnel home or work place in the case of a stand alone system or small network. The recommended setup for the technical support person is based on a PC rather than a terminal, because of its ability of uploading files. A portable computer adds to this the convenience of easy transportation to almost any location.


	<i>HP 3000</i>	<b>SB09*/9</b>
	<i>INTERNATIONAL CONFERENCE</i>	
	<i>VIENNA 1987</i>	

In a large networks discipline has to be used in remote maintenance and operations. Changes to an individual site will have to be reviewed and implemented across the network if appropriate. Also documenting all these activities is essential, this can automated in the case a PC or system based remote maintenance and operations setup.

## **SUMMARY**


I hope I have provided you with some new ideas in the area of operations automation. There are many third party products available that can help in your efforts to develop the right solution for your environment. Local availability of these products may vary and this will affect what the most viable solution is. Companies that have implemented tools to move towards an operatorless environment have experienced the expected savings, but typically have also found that system uptime increases. Another result expressed specifically by the end users was a more predictable "behavior" of the system, which was felt to be a productivity gain.

Coming back to the question, is an operatorless HP3000 a fact or is it fiction, I believe that the answer is determined by reality. Yes, it is possible to fully automate the operations of an HP3000, but the cost aspect dictates that automating only up to a certain point is justified.

	<i>HP 3000</i>	<b><i>SB09*/10</i></b>
	<i>INTERNATIONAL CONFERENCE</i>	
	<i>VIENNA 1987</i>	

## **BIOGRAPHY**

Henk Alblas is currently a member of the Application Engineering Organization of Hewlett-Packard Canada in the Toronto Office. He has been involved in many presales projects and he has provided account management support to a variety of existing HP3000 customers. Henk has been with HP in Canada for over four years and before that he has worked for HP in the Netherlands. Before joining HP he was employed by ICL and Wang both in Europe and in Canada. His hobbies are horseback riding, winter sports and photography.

	HP 3000	DB19*1
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Inter-system Database Interaction in a Networked Environment

David W. MacKay  
Hewlett-Packard (Canada) Ltd.  
Toronto, Ontario, Canada

### Abstract

The scenario of an executive reaching out to a variety of sources to pull together whatever information is desired with just a few keystrokes on a terminal or personal computer is often quite removed from what is possible in reality. What prevents this scenario from being commonplace is not an inability to network different computer systems, but rather an inability to access what information is available on one's own and other systems in a network. Access to information, not interconnectivity, is the problem.

Disparate computer systems, even those manufactured by the same vendor, may employ different, nonhomogeneous, database architectures that make the task of providing for meaningful and efficient inter-system database interaction difficult. This paper explores issues and techniques involved in providing for inter-system database interaction. Topics examined include:

- providing for meaningful communications between systems employing disparate database architectures,
- database administration in a nonhomogeneous distributed database environment, and
- uses of the HP3000 in such an environment with examples of how these difficulties are being overcome.

The usefulness of a network of computer systems is directly related to the user's ability to access the information resident on each system in the network. By providing for meaningful inter-system database interaction, the productivity of all users of a computer network may be increased.

### Introduction

Imagine a computing environment in which several computer systems are linked into a network. For example, our network might consist of an HP3000 running Image, an HP1000 also running Image, an HP9000 running ORACLE, and an HP Vectra with d:BASE II. For the sake of argument, we might also include a VAX running Rdb or INGRES, and an IBM system running IMS or CICS (see Fig. 1). This is a network of nonhomogeneous (i.e. heterogeneous) database management systems. A homogeneous system would be a network in which all nodes were running a copy of the same database management system.

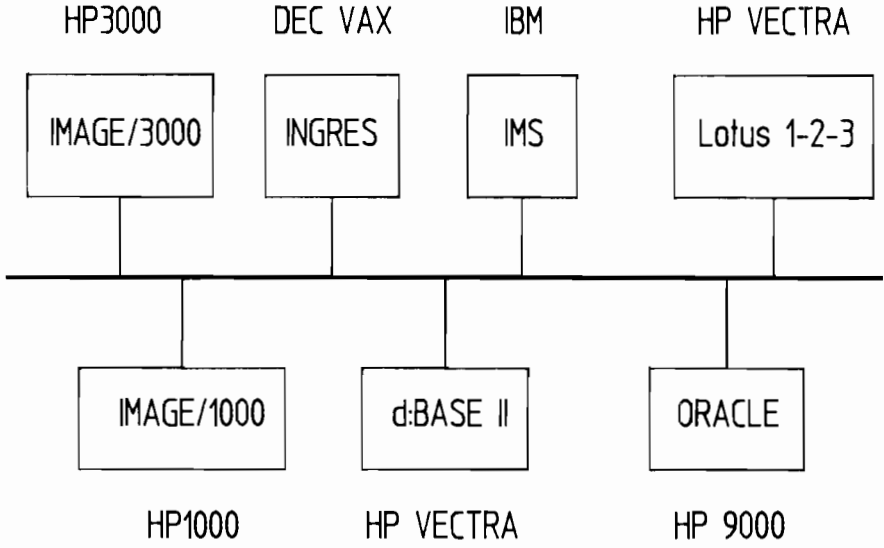



FIGURE 1 : HETEROGENEOUS NETWORK

	HP 3000	DB19'3
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Now picture the poor user who needs to gather information from some, or all, of these systems in order to do his job. We can make the picture even bleaker by requiring the user to process the data he has retrieved in an application on his personal computer, say Lotus 1-2-3. Although a network is in place between the different systems, it is of little help to the user in this situation.


The network is of little help to our user since it is merely a means to link the computers together; it does not provide the services necessary to allow him to access the data resident in each computer's database management system (DBMS). At present, about the best we can hope for a network to provide, when dealing with heterogeneous (i.e. nonhomogeneous) systems, is network file transfer (NFT), and perhaps virtual terminal (VT) capability.

From our perspective, the type of network that is used to link the different computer systems is of little concern. For instance, it should not matter whether the systems are connected via an IEEE 802.3, Ethernet, MAP, or TOP local area network, via an X.25 network, some proprietary network, or whatever. Basically, we can view the network, be it long-haul, local, or a hybrid of the two, as a means by which meaningful inter-system communication might be established. All networks share some characteristics, to greater or lesser degrees, that are of interest to us. Chief among these are that they are slow, have high access delay times, and that they are expensive in terms of CPU instructions per message.

The network services that were mentioned earlier, NFT and VT, are part of the presentation layer (i.e. layer 6) of the ISO's 7-layer Open Systems Interconnection (OSI) Reference Model. The OSI model is useful because it allows us to consider networks as a layer of protocols and a set of services. However, there are not yet any internationally accepted standards for the topmost layers. Of particular interest to us are the session, presentation, and application layers (layers 5, 6, and 7). Hewlett-Packard's own AdvanceNet, for instance, has no specific session layer; layers 5, 6, and 7 are lumped together. This is consistent with the current state of the art in the field of networking. As protocols are developed for the uppermost layers in the OSI Reference Model, we can expect most major computer vendors to add them to their networking products.

It is true that, with network file transfer (NFT) and virtual terminal (VT) capabilities, our user could logon to each of the systems that he needs to access, use each system's DBMS user interface to produce a simple ASCII file of what data is needed, and then transfer each of these files to his local system. He could then manipulate the files, perhaps combining them into a single large file, and then convert that file into one that is acceptable to the application program that he is interested in using, Lotus 1-2-3 in this case.

This sounds like an arduous process, and, without doubt, it would be even more difficult to accomplish than it sounds. Quite apart from the inherent difficulty in the procedure, our user must have a detailed knowledge of each database schema, each DBMS's user interface, the different file formats that he will need to convert files into, and perhaps the topology of the network. Bear in mind that, despite these difficulties, our user can only access one database at a time; he has no facility for concurrently accessing the information that resides in more than a single DBMS. This could present an insurmountable obstacle.

	HP 3000	DB19*4
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Our user needs a better way. He needs to be able to deal with the collection of DBMSs in his network as if he was dealing with a single centralized DBMS. He also needs to be able to have his results formatted so that they are usable by his application programs, such as Lotus 1-2-3. This paper is concerned with what is involved in providing a solution that meets these needs.

## Distributed Database Management System Reference Architecture

Inter-system database interaction in a networked environment is intended to allow for the user of any computer in a network to access whatever data is in the network as easily as if the data were on the user's own system. In other words, we would like to provide a "global", or network-wide, database management system that incorporates all of the local DBMSs on the network's nodes and looks like a centralized DBMS to the user. Such a system is known as a distributed database management system (DDBMS).

A centralized DBMS may be viewed as being comprised of a client processor and a data processor, according to a reference architecture synthesized by Larson and Rahimi, upon which this paper is based. The client processor is responsible for interfacing with a user. It translates user commands into a canonical language that the data processor understands. It also translates data that it receives from the data processor into a format understandable by the user. The data processor is responsible for storing and accessing data in the database. Both client processors and data processors can be subdivided into other processes (see Figures 2 and 3). Figure 4 is a representation of a centralized DBMS.

Larson and Rahimi extend their DBMS architecture to encompass DDBMSs through the addition of a global database control and communications system. Figure 5 is a representation of a DDBMS; Figure 6 shows the processes that comprise a global database control and communications system.

We now proceed to describe each of the five components of the reference architecture synthesized by Larson and Rahimi. These components are canonical structures, a data dictionary, a client processor, a data processor, and a global database control and communication system.

### Canonical Structures

Canonical structures for commands and data are an important element of this, and almost all other proposed reference architectures for DDBMSs. A canonical structure, be it a language or a data format, is simply a standard that is understood by all network nodes that participate in the DDBMS. Homogeneous DDBMSs need not be concerned with implementing canonical structures since there is no conversion required from one local DBMS to any other local DBMS. Performance should be higher when no conversions are necessary, but the difference may not be significant in light of datacomm and other overhead that can be orders of magnitude greater than the cost of converting to and from canonical forms.

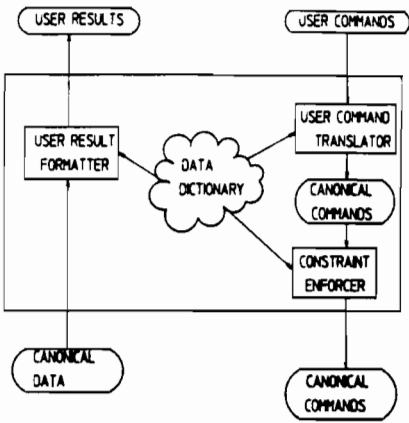


FIGURE 2 : CLIENT PROCESSOR

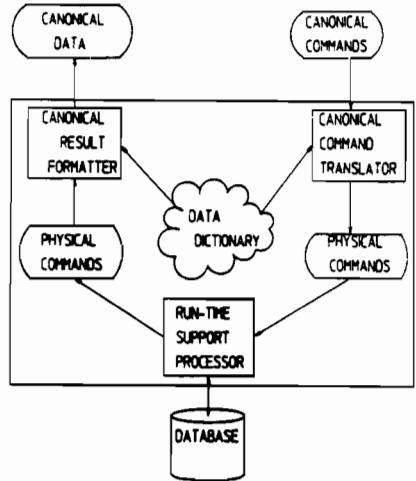


FIGURE 3 : DATA PROCESSOR

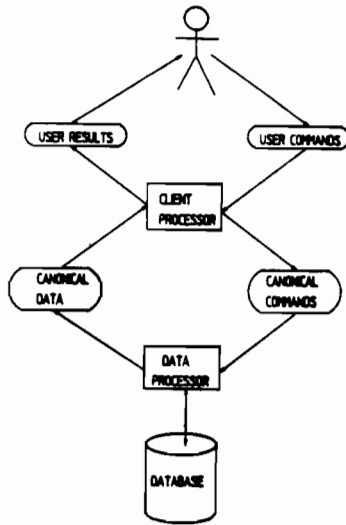


FIGURE 4 : CENTRALIZED DBMS



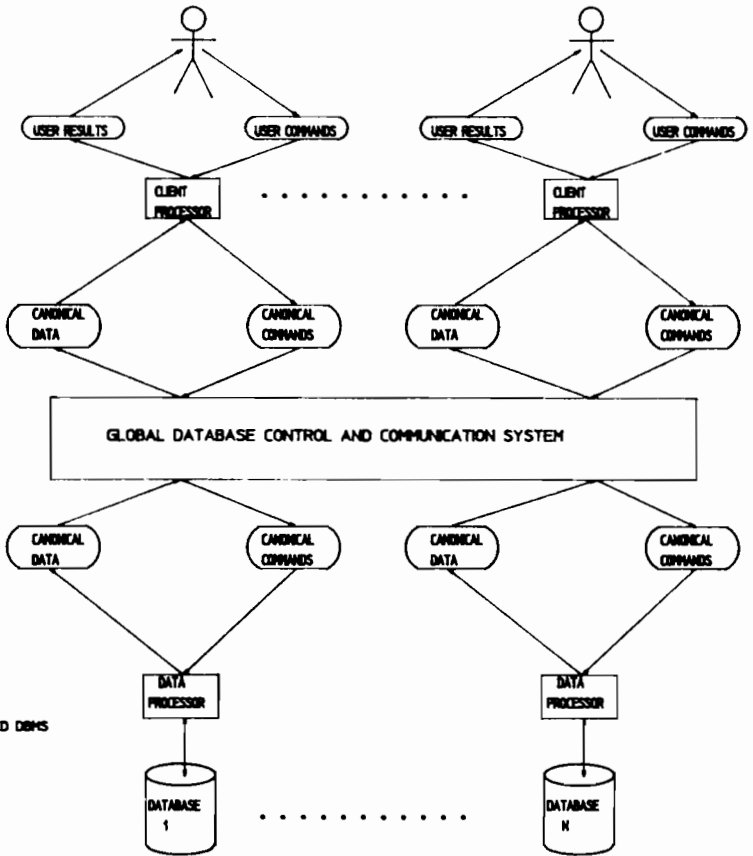



FIGURE 5: DISTRIBUTED DBMS

	HP 3000	DB19*7
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

It is often assumed that the canonical language of choice will be SQL. A nonprocedural language, such as SQL, is a good choice as a canonical language. A nonprocedural language uses high-level commands (i.e. set-at-a-time) so that several records might be selected by a command, as opposed to the single record that a procedural language command might select. This has the effect of reducing communication overhead. Nonprocedural languages view the data in a system via the relational data model. This allows relational operators, such as JOIN, to be used to deal with data partitioning. An example of data partitioning might involve an organization that maintains its personnel records at local offices, and not centrally. The global personnel database would then be the combination of all of the local personnel databases. Furthermore, SQL has recently been adopted as ANSI standard X3H2. Although it is likely that SQL will assume the role of canonical language, it should not be regarded as an ideal choice. It suffers from, among other weaknesses, a lack of orthogonality, and incomplete support for some aspects of the relational model.

Just as SQL is often expected to become the canonical language of choice, "flat" ASCII files are often expected to become the canonical data form most widely used in heterogeneous DDBMSs. ASCII files may be regarded as the lowest common denominator among file types, and that may help to establish "flat" ASCII files as a canonical data form. That same characteristic is also the Achille's heel of ASCII files; they are not able to adequately represent many structures, particularly those involving pointers, variant records, and unusual word sizes.

### Data Dictionary

A key element of this reference architecture is the data dictionary. The data dictionary is used to store the global schema of the DDBMS, information on what data is where, information useful to control and administer the DDBMS, and other information. It may contain a copy of the schemas for each local DBMS in the DDBMS. The DDBMS uses the information in the data dictionary when accessing the database. The data dictionary is stored as a database so that it can be interrogated with normal DBMS interfaces, and also so that it can take advantage of the concurrency control provided by the global database control and communication system. Issues involving the data dictionary concern how it is stored in the DDBMS, which affects both DDBMS performance and the autonomy of the local DBMSs that comprise the DDBMS. A DDBMS with a centralized data dictionary offers no local autonomy since all requests, even those for data in the local DBMS, must be routed through the central site. If all sites in the network have their own copies (replicas) of the data dictionary, the system is considered to be fully replicated. In this case, the task of ensuring replication consistency may consume an inordinate amount of the network's bandwidth and thereby offset the performance improvement gained by having a copy of the data dictionary at each local DBMS. Replication consistency is the responsibility of the distributed execution monitor.

## Client Processor

The client processor is composed of a user command translator, a user result formatter and a constraint enforcer.

User commands might be entered in the form of QUERY/3000 or QUEL commands, or commands from any of a variety of other query languages, be they interactive or programmatic. Different query languages are written for DBMSs that use a particular database model. As an example, QUERY/3000 was written for the HP Image network DBMS. Similarly, QUEL was developed for use with the INGRES relational DBMS. Although no general solution is known (partly because most DBMSs do not adhere to strictly defined database models), it is usually possible to provide a mapping between hierarchical, network, and relational database models. The use of a canonical command language facilitates mapping between different database models.

Converting all user commands into a canonical language allows new query languages, and therefore new local DBMSs, to be easily incorporated into a DDBMS by reducing the amount of effort required to provide command conversion routines. For  $n$  different user command languages in a DDBMS that employs a canonical command language there are  $2n$  conversion routines (i.e. a user command translator, and a canonical translator for each different command language), whereas without a canonical language,  $n(n-1)$  conversion routines need to be written (eg. A to B, A to C, B to A, B to C, etc.).

The constraint enforcer guarantees that only data that meets the semantic integrity constraints specified by the data dictionary is allowed into the database. This serves as an alternative to having update programs determine whether or not the values of data are valid.

The user result formatter converts the data that is returned as a result of the user's commands into a useful format, such as a report, or a .WKS file for Lotus 1-2-3, for example. The use of a canonical data format facilitates conversion to many different file formats.

## Data Processor

The data processor is composed of a canonical command translator, a canonical result formatter, and a run-time support processor.

The canonical command translator converts the canonical commands that were generated by the user command translator into physical commands that the run-time support processor will execute. If the target DBMS was IMAGE/3000, and the user submitted a query using SQL commands, the client processor would translate the SQL commands into a canonical language, call it ANYLANG. The Canonical command translator would then translate the ANYLANG commands into QUERY/3000 for use by the run-time processor.

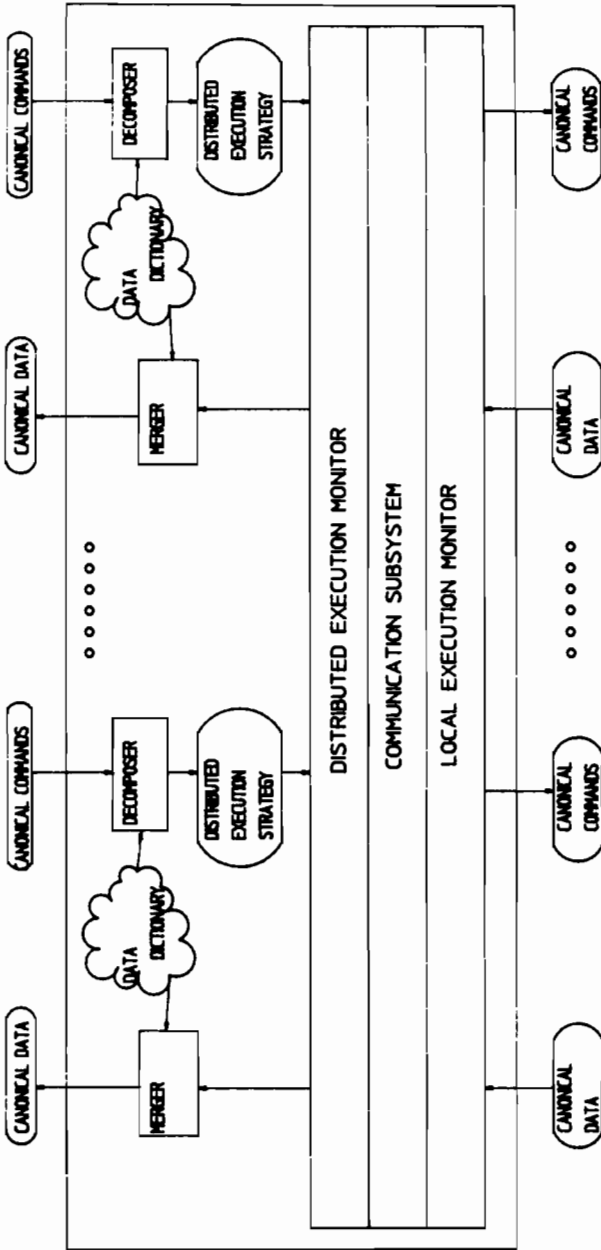



FIGURE 6 : GLOBAL DATABASE CONTROL AND COMMUNICATION SYSTEM

	HP 3000	<b>DB19*/10</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The run-time support processor is responsible for interacting with the local DBMS. In fact, the run-time support processor will normally be the standard user or programmatic interface that comes with the local DBMS. The fact that it does not require any modification in order to be incorporated into the DDBMS is a significant feature of this reference architecture.

The canonical result formatter accepts data from the run-time support processor and converts it into a canonical form.

### Global Database Control and Communication Subsystem

The global database control and communication system is composed of six components: the decomposer, the distributed execution strategy, the merger, the distributed execution monitor, the communication subsystem, and the local execution monitor. It is the element of the reference architecture that allows for a group of individual DBMSs to be joined to form a DDBMS. Accordingly, we shall examine its components in some detail.

#### 1. Decomposer

The decomposer is responsible for translating a global request for information into a distribution execution strategy. It is possible that the data requested by a transaction might reside at several different sites in the DDBMS. The decomposer consults the schema information in the data dictionary to find which local DBMSs in the DDBMS have what data. It then decomposes the transaction into subtransactions to be executed by local DBMSs within the DDBMS. Deciding which local DBMSs should do what, and when, is an important responsibility of the decomposer, and is known as query optimization.

The importance of query optimization to the performance of a DDBMS cannot be overstated. Date provides an example where, just by varying the optimization strategy used by the decomposer, the time taken by the DDBMS to complete a transaction can vary from 1 second to 2.3 days.

#### 2. Distributed Execution Strategy

The distributed execution strategy is the output of the decomposer. It is essentially an ordered list of which local DBMS will execute what subtransaction.

#### 3. Merger

The merger works in a fashion complementary to that of the decomposer. The decomposer disassembles a transaction into a list of subtransactions; the merger takes the results of the execution of the subtransactions and combines them appropriately before passing the data to the client processor.

#### 4. Distributed Execution Monitor

The distributed execution monitor is concerned primarily with the enforcement of three requirements of a DDBMS: transaction atomicity, concurrency control, and replication independence. Ensuring that these three requirements are enforced is perhaps the most difficult task of a DDBMS. Accordingly, we deal with them at length.

##### a) Transaction Atomicity

The concept of a transaction was developed, according to Lindsay, to "delimit the sequences of actions which together transform the database from one consistent state to another". Transaction atomicity is a property common to all database systems in that transactions always leave the database in such a state that transactions appear to either execute to completion, or not at all. The distributed execution monitor supports transaction atomicity at a global level. The decomposer splits a global transaction into a number of subtransactions which are executed by local DBMSs in the DDBMS, under the supervision of local execution monitors. Each local execution monitor reports to the distributed execution monitor of the success or failure of the subtransaction that it supervises. Unless all local execution monitors report success, the global transaction has failed. The distributed execution monitor keeps track of the success/failure reports of the local execution monitors. If all local execution monitors report successful completion, the transaction is committed (i.e. made permanent), otherwise the transaction and all subtransactions are aborted, and do not change any databases.

##### b) Concurrency Control

Concurrency control attempts to keep a database internally consistent by preventing the concurrent execution of conflicting transactions. Several ways to ensure internal consistency have been suggested, but the two most well known methods are locking and timestamping. Either of these methods may be used to prevent two or more transactions from interfering with each other by accessing the same data object, or resource, concurrently.

When locking is used to provide concurrency control, as is the case with DBMSs such as HP Image and HP ALLBASE, a transaction attempting to reference a data object must first attempt to lock it. If no other transaction has secured a lock on the object, the transaction is granted a lock, and now has exclusive access to the object; no other transaction may reference the object until the lock is released by the transaction holding it. If a transaction requests a lock on an object that is already locked, the requesting transaction may be suspended until the lock is released. Actually, locking allows for more sophisticated abilities than those described above. Many DBMS, such as HP ALLBASE, allow several types of shared locks and locking granularities. (Granularity is a term used to describe the size of an object being locked. Record locking is of finer granularity than file locking, for example). Nevertheless, the motivation for locking is to limit access to a data object to the first transaction that requests it.

Locking has one major drawback, that being that conditions known as deadlocks are possible. A deadlock arises when two or more transactions are suspended waiting for locks held by the other suspended transactions. As an example, imagine two transactions, T1 and T2, and two data objects, A and B. T1 has an exclusive lock on A and T2 has an exclusive lock on B. T1 requests a lock on B and is suspended. T2 requests a lock on A and is also suspended. Both T1 and T2 will stay suspended forever if the DBMS does not take action to resolve the deadlock (usually by aborting either T1 or T2). Deadlock detection and prevention algorithms are well-established for DBMSs. Unfortunately, deadlock detection and prevention is significantly more difficult for a DDBMS than it is for a DBMS.


Timestamping is another method by which database consistency may be assured. It is an alternative to locking, but they are not exclusive of each other; they could be used together to provide a hierarchy of concurrency control mechanisms. A system employing timestamping to ensure internal consistency assigns to each transaction an identifier comprised of a timestamp and the DBMS's network node address. The node address is necessary to prevent any transactions from having the same timestamp. This unique identifier gives an indication of serial execution order; the transaction with the oldest timestamp is awarded access. Locking is susceptible to deadlock, timestamping is not. Problems associated with timestamps involve clock synchronization.

Both locking and timestamping are effective in keeping a database internally consistent by preventing the concurrent execution of conflicting transactions. We have seen how the distributed execution monitor ensures transaction atomicity, and provides for concurrency control. The distributed execution monitor is also responsible for keeping a database mutually consistent through replication independence.

### c) Replication Independence.

Replication involves having one or more copies (replicas) of the same data object in the DDBMS. Replication may improve the availability and reliability of a DDBMS since data that might otherwise be unavailable to the DDBMS (if a DBMS were to crash, for instance) could still be available if the data object had been replicated. We can reduce communications costs by replicating frequently accessed data objects nearer to the DBMS that requests the data. Of course, if we were to replicate the data object at the local DBMS, communications cost would be effectively eliminated. Replication involves the ability to distribute, and manage, copies of data objects throughout the DDBMS.

A monthly price list would be a good candidate for replication so that more than one, if not all, of the DBMSs in a DDBMS could have their own copy. The data dictionary would instruct the decomposer to direct all requests for price list information to the copy nearest the requester. This could improve performance markedly, particularly if the user had a copy resident at his local DBMS. The fact that he is accessing a copy would be transparent to the user. Indeed, as long as the data in every price list was mutually consistent with the data in every other price list (i.e. all are identical), there is no difference between the original price list and the replicated versions. If replicated data objects are to be useful, we must ensure mutual consistency.

	HP 3000	<b>DB19*/13</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Note that we have provided for access to replicated copies of a data object to be transparent. We recall that our goal is to provide the user of a DDBMS with the illusion that he is using a single centralized DBMS. We want to enable the user to ask for data and not need to be concerned with where the data is located in the DDBMS. We can meet our goal, as it relates to replicated data objects, by ensuring that two conditions are met: mutual consistency and transparent data access.

Unfortunately, replication is best suited for read-only data. Imagine a user wanting to update the information in the price list. Keep in mind that it is only necessary to access any one of the replicated data objects to perform a read, but that a write must update them all. If all copies are not updated, the copies are no longer mutually consistent. The limited speed of the network prevents all replicas from being updated simultaneously, so that, despite our best efforts, there will always be times when all copies are not yet mutually consistent. Our goal, then, must be to ensure that all of the replicated data objects converge to consistency as soon as possible after a transaction has been committed. This is known as update propagation.

Update propagation is concerned with the task of ensuring mutual consistency among replicated data objects. Individual DBMSs within a DDBMS may be unavailable for a variety of reasons, such as system failures. This, as well as the problem caused by communication delays, makes the problem of update propagation difficult to solve. The techniques that are used to ensure internal consistency, such as having the complete transaction fail if any single subtransaction fails, are not applicable here. It would be unreasonable to roll back transactions because replicated data objects were unavailable.

If there are 10 copies of a data object in a DDBMS where individual DBMSs are available 99% of the time, we find that the probability of an update succeeding is only 90%. We do not want to have to roll back 10% of our transactions, so we must find a more attractive means of ensuring mutual consistency.

Three approaches have been suggested as solutions to the problem of update propagation: transaction spooling, primary/secondary copy relationships, and snapshots.

Transaction spooling techniques involve keeping lists of what updates need to be applied to each replicated data object that is unavailable. Variations on this technique have the lists themselves being replicated in case the list-keeping DBMS should become unavailable. When the DBMS is once again able to resume normal operation it must first go through a recovery phase where it applies the updates to its local data objects. Once this recovery phase has been completed, the DBMS may resume normal operation.

An approach involving the establishment of primary/secondary data object relationships requires all updates to be directed to the primary data object first. The primary data object is then responsible for applying the update to its secondary data objects. This procedure is to be followed even when all of the DBMSs containing the secondary data objects are available. An improvement to this approach involves circulating the primary-copy responsibility among the mutually consistent data objects. Provisions need to be made to ensure that accesses are only directed to updated (i.e. mutually consistent) secondary data objects.



A completely different approach to update propagation involves the use of "snapshots". Snapshots are copies of a data object, as it existed at a certain point in time. Snapshots are never updated, although periodically they may be replaced with a more recent snapshot. Since updates are not allowed, the problem of update propagation does not need to be addressed. All snapshots are of a single master data object, to which all updates are directed. The price list example mentioned earlier might lend itself well to a solution involving the use of snapshots. Snapshots do not allow for replication transparency, since all users must be aware that they are working with data that was accurate only "as-of" a certain point in time.

Replication independence is intended to allow the DDBMS to place copies of data in a number of local DBMSs throughout the DDBMS. We can only attain our goal of providing a user with the illusion that he is working with a single centralized DBMS if replications are mutually consistent and their use is completely transparent to the user. The problem of update propagation makes this difficult to accomplish.


#### Distributed Execution Monitor --- Summary

Our examination of the function of the distributed execution monitor subsystem within the global database control and communication system has been lengthy enough that a summary is in order. The distributed execution monitor is responsible for addressing three areas of crucial importance and considerable difficulty. It ensures global transaction atomicity by working closely with the local execution monitors that supervise each of the subtransactions that comprise a transaction in a DDBMS. It is responsible for ensuring internal consistency by preventing the concurrent execution of conflicting transactions. Two methods for accomplishing this are locking and timestamping. It is also responsible for ensuring mutual consistency among the copies of a data object that might exist in a DDBMS, so that the existence of these copies is transparent to users. Update propagation is the greatest problem in providing for mutual consistency.

#### 5. Communication Subsystem

The communication subsystem is used to transfer requests and data between sites. The 7-layer OSI model makes provision for a number of features that could be useful in a DDBMS. For instance, part of the functionality of the session layer (layer 5) is dialogue control. Tanenbaum describes dialogue control as

... bracketing groups of messages into atomic units. In many database applications it is highly undesirable that a transaction be broken off part way, as a result of a network failure, for example. If the transaction consists of a group of messages, the session layer could make sure that the entire group had been successfully received at the destination before even attempting to start the transaction.

	HP 3000	DB19*/15
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Tanenbaum goes on, however, to bemoan the fact that few networks have implemented this, presumably due to the lack of well-accepted standards for levels 5, 6, and 7 in the OSI Reference Model. Other network services, such as file transfer (NFT) are desirable, but can be provided by user programs if such services are not provided by the network.

#### 6. Local Execution Monitor

The local execution monitor is responsible for the execution of the subtransaction at its node. It notifies the distributed execution monitor as to whether the subtransaction succeeded or failed. It may be seen as being an extension to the run-time support processor.

#### Global Database Control and Communication --- Summary

The global database control and communication system is the subsystem that differentiates a DDBMS from a DBMS in the reference architecture synthesized by Larson and Rahimi. It provides the means to link multiple heterogeneous DBMSs into a network so that a user is presented with the illusion that he is working with a single, centralized DBMS.

#### DDBMS Reference Architecture --- Summary

We view a distributed database management system (DDBMS) as being a suitable means for achieving the inter-system database interaction that is our goal. A DDBMS may be considered to be a database spanning the nodes in a network and comprised of the DBMSs that are resident at the different nodes in a network.

A reference architecture has been described that views a centralized DBMS as being comprised of a client processor, and a data processor. We are able to connect such centralized DBMSs together so as to form a DDBMS through the addition of a global database control and communication system. Canonical structures are used to facilitate communication between heterogeneous local DBMS in the DDBMS. That is, all inter-system communication, facilitated by the global database control and communication system, takes place in the form of canonical structures for both commands and data. A data dictionary is essential to the operation of each of the subsystems that comprise a DDBMS (i.e. client processor, data processor, and global database control and communication system). The data dictionary acts as a roadmap to all of the data within the DDBMS. These five elements can work together to provide for meaningful and efficient inter-system database interaction among multiple nonhomogeneous DBMSs in a network.

## Database Administration in a Nonhomogeneous DDBMS Environment


In order to manage a DDBMS it is necessary to share data administration authority among several individuals. This is necessary because management of a DDBMS involves more than merely managing the operations of each DBMS in the DDBMS. That will, of course, still need to be done, but other tasks will be required of the managers of each individual DBMS as well. Additionally, a global administrator will be needed. This individual must assume responsibility for any tasks that do not clearly belong to any single DBMS's database administrator (DBA). The responsibility for managing a DDBMS belongs to both the global DBA and the local DBAs.

We briefly outline some of the tasks involved in administering a DDBMS, as listed by Walker. This list is not intended to be comprehensive.

### Activities involved in the administration of a DDBMS:

1. **Communication:** The lines of communication between users, management, operations, support, and maintenance groups must be kept open.
2. **Monitoring:** Hardware and software must be monitored. This involves more than just failure tracking. It also involves planning and coordinating upgrades and replacements.
3. **Testing:** It is necessary to test for user acceptance, data accuracy, and database integrity, for example.
4. **Training:** It is necessary to provide training for users and DDBMS staff.
5. **Scheduling:** It is necessary to schedule tasks, personnel, repairs, etc. to provide for maximal DDBMS utilization.
6. **Documentation:** It is necessary to provide users with adequate documentation as well as to maintain up-to-date documentation sets for any applications.
7. **Accounting:** Some means must be devised to charge users for the resources that they consume.
8. **Performance:** It is necessary to assign priorities to tuning activities, in an attempt to maximize system performance.
9. **Design:** Existing applications and database schemas will require periodic review. New applications may need to be designed.
10. **Operations:** It is, of course, necessary to ensure that the operations of each DBMS proceed smoothly.
11. **Backup and Recovery:** It is necessary to design and test a backup and recovery procedure.
12. **Security:** Data must be protected against unauthorized access and modification.

The local DBA is responsible for the data that resides on the system he administers. It is natural, then, for him to insist that the DDBMS not reduce the autonomy of his local DBMS.

	HP 3000	DB19*/17
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The global DBA will be concerned with coordinating the operation of the local DBMSs in the DDBMS. He will be responsible for ensuring that standards are developed and strictly adhered to. He may also be responsible for ensuring that the communication subsystem (i.e. the network) functions correctly, although this task would be better handled by a network administrator. The design of the data dictionary and canonical structures might also be his responsibility.

Although the administration of a DDBMS can be quite complex, we may simplify our view of it by dividing the administration of a DDBMS into two areas of responsibility. Each local DBMS DBA may be regarded as being responsible for all tasks that are totally within the realm of each respective local DBMS. Global tasks include the coordination of all local DBA tasks and are the responsibility of the global DBA.


The administration of a DDBMS is the responsibility of a number of individuals, namely the DBAs responsible for local DBMSs and the DBA responsible for the global DDBMS. The effort involved in managing a DDBMS is greater than the sum of the efforts involved in the management of each of the local DBMSs that comprise the DDBMS. Local DBAs will be primarily concerned with the autonomy and successful operation of the DBMSs for which they are responsible. The global DBA is responsible for, among other things, coordinating the operation of the individual DBMSs that comprise the DDBMS.



#### Uses of the HP3000 in a DDBMS Environment

Heiler and Maness are correct when they say that, with regard to the difficulties associated with heterogeneous DDBMSs "we are far from a solution to these problems". Indeed, before heterogeneous DDBMSs, of the type we described earlier, become widely available, a number of problems, such as those involving update propagation, need to be solved. The success of heterogeneous DDBMSs will depend, to a large degree, on the availability of standards. Of particular importance are a standard reference architecture, and standards for canonical structures. It is our hope that a standard reference model will do for DDBMSs what the ISO OSI model has done for computer networks. Until such standards are developed, heterogeneous DDBMSs must, of necessity, employ proprietary architectures. It is doubtful that such proprietary DDBMSs would be sufficiently flexible to accommodate a great variety of different computer systems and DBMSs. It will probably be five to ten years before nonhomogeneous DDBMSs, as described in this paper, are available.

Of course, efforts are underway to provide some of the functionality that we have described. Two approaches have attracted considerable attention. The first involves adding network support to homogeneous DBMSs. The other approach involves a very limited form of nonhomogeneous DDBMS functionality, which we will term "heterogeneous integration" to clearly distinguish it from the DDBMSs that we have described elsewhere in this paper. We discuss each of these approaches separately.

	HP 3000	DB19*/18
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## 1. Networked homogeneous DBMSs

The two most vocal proponents of networked homogeneous DBMSs are Oracle, for their ORACLE DBMS, and Relational Technology Inc., for their INGRES DBMS. The two companies have extended their core DBMSs by offering both a network subsystem and a distributed database subsystem. ORACLE terms their subsystems SQL\*Net and Distributed ORACLE; the corresponding INGRES products are INGRES/Net and INGRES/Star. Both ORACLE and INGRES are available on a wide range of host computer systems, so that the databases are heterogeneous in that different manufacturer's computers may be included in the networked homogeneous DBMS, as long as all systems are running the same DBMS. Both companies offer "gateways" to allow for non-ORACLE or non-INGRES DBMSs to be included in the networked homogeneous DBMS. Systems connected via gateways have limited functionality. Both ORACLE and INGRES are able to include personal computers in their respective networked homogeneous DBMSs. Versions of each DBMS are available for PC AT compatible computers, such as the HP Vectra. The two DBMSs differ in their approach as to how they support data formatting for popular PC (and other) applications. Recall that, in the reference architecture presented earlier, the user result formatter allowed us to format data for use by different popular applications, such as Lotus 1-2-3.


ORACLE can import data from many popular PC applications into their SQL\*Calc subsystem and from there into the database directly. They can export data only in the form of ASCII files. They feel that there is significant benefit to be had by working with applications that are more closely linked to the DBMS than is, say, Lotus 1-2-3. For this reason they offer a number of alternatives to popular PC applications.

INGRES, on the other hand, provides functionality similar to that provided by the user result formatter, via their INGRES/PCLINK subsystem. They are able to input and output data in a wide variety of formats, suitable for use by most popular PC application packages.

ORACLE and INGRES are the industry leaders in providing networked homogeneous DBMSs. Hewlett-Packard customers are able to choose which of these two networked homogeneous DBMSs is most suitable for their needs; both ORACLE and INGRES are available on the HP9000 family of computers.

## 2. Heterogeneous Integration

Hewlett-Packard is the industry leader in providing integrated heterogeneous systems. The vehicle that best illustrates our abilities in this area is Hewlett-Packard's Personal Productivity Centre (PPC). Heterogeneous integration is provided by a set of four subsystems known collectively as HP's Office Productivity Services. The four components in the Office Productivity Services are HP AdvanceNet networking, Resource Sharing, Information Access, and HP DeskManager. Of these, AdvanceNet networking, Resource sharing, and, most of all, Information Access, are of interest to us. The functionality provided by HP's Office Productivity Services is unmatched in the industry.

	HP 3000	DB19*/19
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

The networking components of HP's Office Productivity Services adhere to HP's AdvanceNet networking architecture. AdvanceNet is based on the OSI Reference Model. HP is firmly committed to standards in networking. The following networks are supported via AdvanceNet: ThinLAN, StarLAN, Serial Network, and RS-232C. All are based on industry standards and may be used by either Resource Sharing and Information Access. Among the computer systems tightly integrated with these networks are the HP3000, HP Vectra, HP Touchscreen, and IBM PC family of computers. Gateways to other computers are available off-the-shelf so that users can expand their network to include computers from other manufacturers. No other vendor offers this level of office functionality across such a broad range of networking choices.

Resource Sharing provides truly transparent shared access to files, printers, applications, and other devices. Disk sharing improves communications, and data security. It also allows PC data to be consolidated where it can be shared, controlled, and secured. Disk sharing allows PC users the benefits of enormous amounts of HP3000 disk storage with throughput that is greater than that of floppy disks. Since the MS-NET defacto standard is fully supported, the system can provide complete MS-DOS 3.X file sharing and locking capabilities for full support of multi-user applications. Resource Sharing allows for PC users to back up their disks to large disks or tape drives elsewhere in the network.

Information Access is the component of HP's Office Productivity Services that provides for inter-system database interaction. It provides end users with the ability to query multiple heterogeneous DBMSs through a simple, menu-driven user interface without requiring the user to know where the desired information is located. Data can be automatically and transparently formatted into (and from) a variety of formats suitable for use with most popular PC applications, such as Lotus 1-2-3. Data in heterogeneous databases, such as d:BASE II and R:BASE 5000, can be reformatted into other database formats, and back again. All of this is possible while high security controls are maintained. For example, in addition to standard Image/3000 and MPE security, the HP3000 DBA determines specific Image/3000 databases, datasets, and even specific data items that can be accessed. This degree of granularity is built on top of standard Image and MPE security by Information Access. This rich complement of features, and others, make it the industry leader. According to criteria established by InformationWEEK, Information Access could be classified as an "ideal product".

It is not surprising that Information Access has a number of features in common with aspects of the DDBMS reference architecture with which we are familiar. After all, both Information Access and some subsystems of the reference architecture were designed to address some common needs. It would be misleading to map Information Access onto the DDBMS reference architecture; the two were developed entirely independently and all that we would actually achieve would be to demonstrate a commonality of features. It seems presumptuous to position Information Access, as it exists at this point in time, as a suitable base upon which to build a DDBMS that would satisfy the requirements of our reference architecture. It is, however, a suitable, if not even a key component, of networks that exhibit heterogeneous integration.

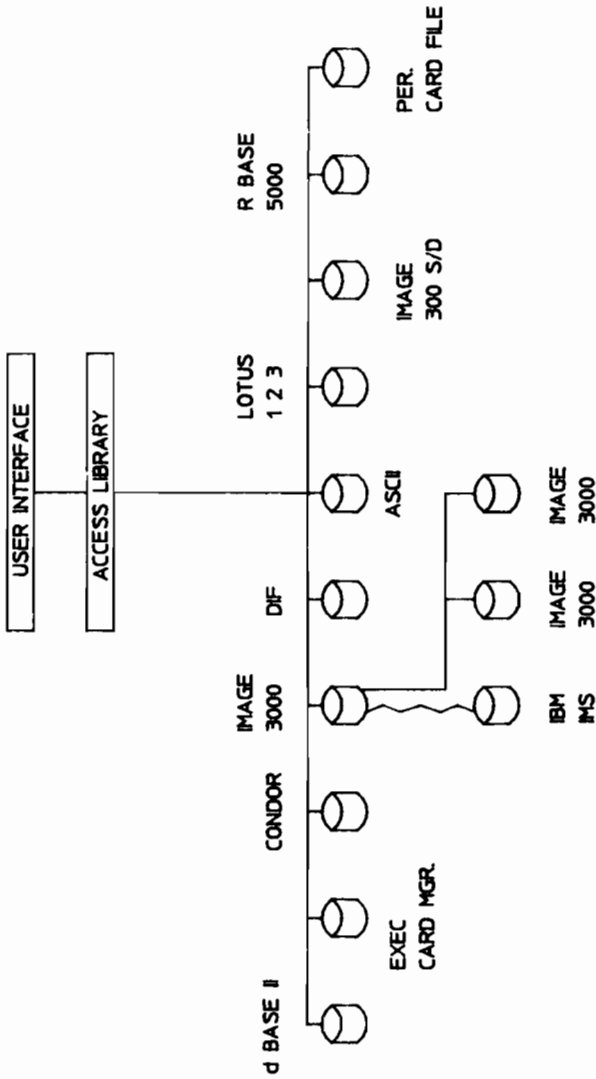



FIGURE #7 CONCEPTUAL VIEW OF INFORMATION ACCESS

	HP 3000	DB19*21
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

Information Access may be considered to be comprised of three tiers: the user interface, an access library, and a tier consisting of a variety of method files (see Fig. 7).

The Information Access user interface provides an alternative user interface to the one provided by a data source. (We define a data source to be any DBMS or file structure supported by Information Access). This menu-driven interface provides a relational view of the data selected from one or more data sources. The relational data model was chosen because it is less difficult to convert data from different data models to a relational model than it is to convert to any other data model. Also, the relational model lends itself well to partitioned data via the JOIN operator which enables a user to join two or more databases so that they appear, to the user, to be a single relation. Several data sources may be JOINed into a single table and accessed simultaneously. When a query or other transaction has been defined, the user interface invokes the access library.

The access library consists of about two dozen calls that are supported, via method files, by every data source. These calls are directed to the appropriate method file where they are translated into commands that are specific to the target data source. This technique avoids the overhead of translations to and from a canonical structure, which was considered important for reasons of performance. The access library treats HP3000 systems as a special case. Instead of having the method file generate Image/3000 intrinsics, it instead generates commands in a proprietary nonprocedural language. These commands are then sent to the target HP3000 where a companion program to Information Access is running. This companion program is responsible for providing the relational view of the Image/3000 database that Information Access requires.


Treating the HP3000 in this fashion facilitates the security features that Information Access provides on top of MPE and Image. In other words, this allows for Information Access to implement security at both the personal computer and the HP3000, in addition to the security features provided by MPE and Image. The use of a nonprocedural language allows for high-level (i.e. set-at-a-time) commands, which, in turn results in less datacomm overhead.

The method files may be regarded as being conversion routines. Adding support for a new data source involves only the addition of a method file for the new data source.

Information Access can accept data from Image/3000 databases, R:BASE 5000, d:BASE II, Personal Card File (PCF), Executive Card Manager (ECM), and CONDOR. It can produce output in files compatible with any of the above, except for HP3000 Image. It can also output HP 3000 SD files, .WKS (for Lotus 1-2-3), DIF, and ASCII files.

As is evident from our discussion of Information Access, it is a very powerful tool; it provides for significant inter-system database interaction.



	HP 3000	DB19*/22
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## Uses of the HP3000 in a DDBMS Environment --- Summary

Hewlett-Packard is at the leading edge of technology in providing integrated heterogeneous systems which attempt to provide some of the functionality that the DDBMSs of the future will offer. HP's Personal Productivity Centre is a showcase that positions HP as the industry leader in providing integrated heterogeneous systems that meet real customer needs. The use of personal computers with HP3000 computer systems is the cornerstone of HP's office products strategy. This ensures that the HP3000 will be the vehicle of choice when it comes to providing for heterogeneous integrated systems now and in the years to come.

### Conclusions


A distributed database management system (DDBMS) provides a means whereby we may achieve our goal of providing for inter-system database interaction. We contend that the objective of inter-system database interaction is to allow a user to access whatever data is in the network as easily as if it were on the user's own system; the DDBMS should present a user with the illusion that he is using a single centralized DBMS. The reference architecture that we present distinguishes between a centralized DBMS and a DDBMS on the basis of a global database control and communication system, the use of canonical structures, and an expanded data dictionary.

Unfortunately DDBMSs are fraught with problems, such as update propagation and a lack of standards, for which no solutions are clear. Accordingly, we do not expect DDBMSs to be readily available for five to ten years.

The need for functionality akin to what might be provided by a DDBMS is real. Two approaches are being used to provide solutions to meet these needs: networked homogeneous DBMSs, and systems providing for heterogeneous integration. HP's Personal Productivity Centre is an example of an integrated heterogeneous system.

Hewlett-Packard's strategy with respect to integrated heterogeneous systems in office environments relies on the integration of personal computers and HP3000 computers. Communication between systems will be facilitated by HP AdvanceNet which is a networking strategy based upon industry standards. The HP3000 will be the vehicle of choice for providing integrated heterogeneous systems.

Systems such as HP's PPC and its constituent components, such as HP AdvanceNet and Information Access provide powerful capabilities that allow for rudimentary inter-system database interaction in a networked environment. The powerful capabilities that they provide can allow for "the scenario of an executive reaching out to a variety of sources to pull together whatever information is desired with just a few keystrokes on a terminal or personal computer".

	HP 3000	<b>DB19*/23</b>
	INTERNATIONAL CONFERENCE	
	VIENNA 1987	

## References

- Brown, A.S. et al, "Data Base Management for HP Precision Architecture Computers", in Hewlett-Packard Journal, Vol. 37, No. 12, December 1986.
- Carlson, R.J. et al, "HP AdvanceNet: A Growth-Oriented Computer Networking Architectural Strategy", in Hewlett-Packard Journal, Vol. 37, No. 10, October 1986.
- Date, C.J., An Introduction to Database Systems. Vols. I & II. Addison-Wesley, 1986
- Date, C.J., "A Critique of the SQL Database Language", in SIGMOD Record. Vol. 14, No.3, November 1984.
- Deen, S.M., "Distributed Data Bases: Some Problems", in Proc. of the International Conference on Data Bases: University of Aberdeen, July 1980, ed. S.M. Deen and P. Hammersley. Heydon and Son Ltd., 1980.
- Gligor, V.D. and R. Popescu-Zeletin, "Concurrency Control Issues in Distributed Heterogeneous Database Management Systems", in Tutorial: Distributed Database Management, ed. J.A. Larson and S. Rahimi. IEEE Computer Society Press, 1985.
- Gligor, V.D. and G.L. Luckenbaugh, "Interconnecting Heterogeneous Database Management Systems" in Computer, January 1984.
- Heiler, S. and A.T. Maness, "Connecting Heterogeneous Systems and Data Sources", in Database Engineering Bulletin, Vol. 7, No. 1, March 1984.
- Larson, J.A. and Saeed Rahimi (eds.), Tutorial: Distributed Database Management. IEEE Computer Society Press, 1985.
- Lindsay, B., "Distributed Transaction Management" in Distributed Database. Online Publications Ltd., 1981.
- Rusinkiewicz, M. and Bogdan Czejdo, "Query Transformation in Heterogeneous Distributed Database Systems", in Proc. 5th International Conference on Distributed Computing Systems, 1985. IEEE Computer Society Press, 1985.
- Saykally, D., "PC-Mainframe Links Continue To Confuse", in InformationWEEK, September 22 1986.
- Tanenbaum, A.S., "Network Protocols", in Computing Surveys, Vol. 13, No. 4, December 1981.
- Walker, H.M., "Administering a Distributed Data Base Management System", in SIGMOD Record, Vol. 12, No. 3, April 1982.



HP 3000  
INTERNATIONAL CONFERENCE  
VIENNA 1987