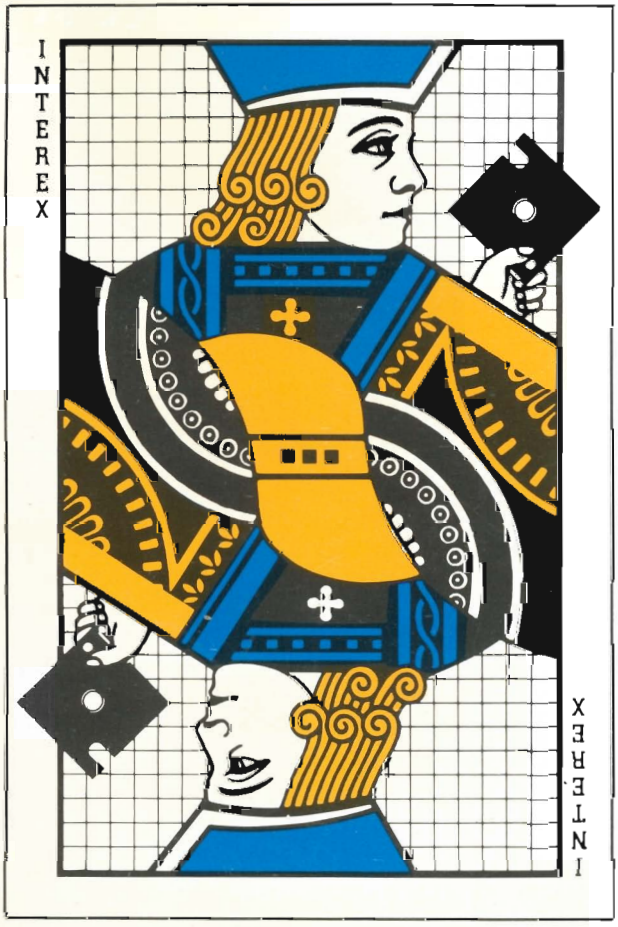
INTEREX



HP BUSINESS USERS
CONFERENCE PROCEEDINGS
VOLUME 3

LAS VEGAS

SEPTEMBER 20-25, 1987

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

INTEREX

the International Association of
Hewlett-Packard Computer Users

Proceedings

of the

1987 North American Conference
Of Hewlett-Packard Business
Computer Users

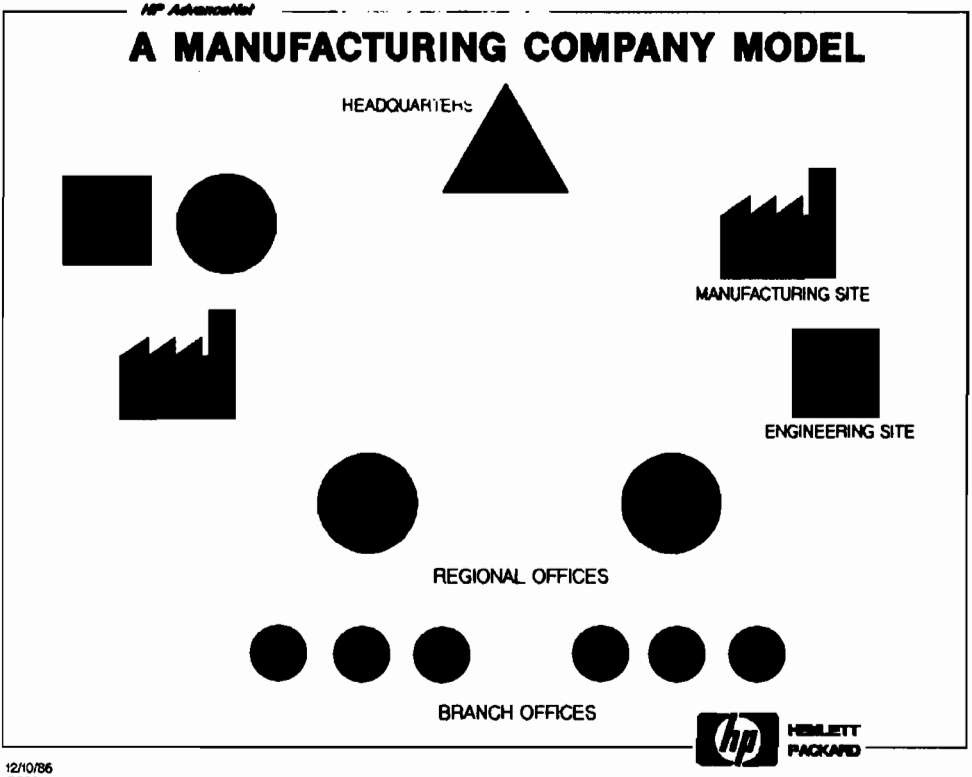
at

Las Vegas, Nevada
September 20-25, 1987

VOLUME 3

BUSINESS OFFICE NETWORKING SOLUTION

STEVE RICHARDSON

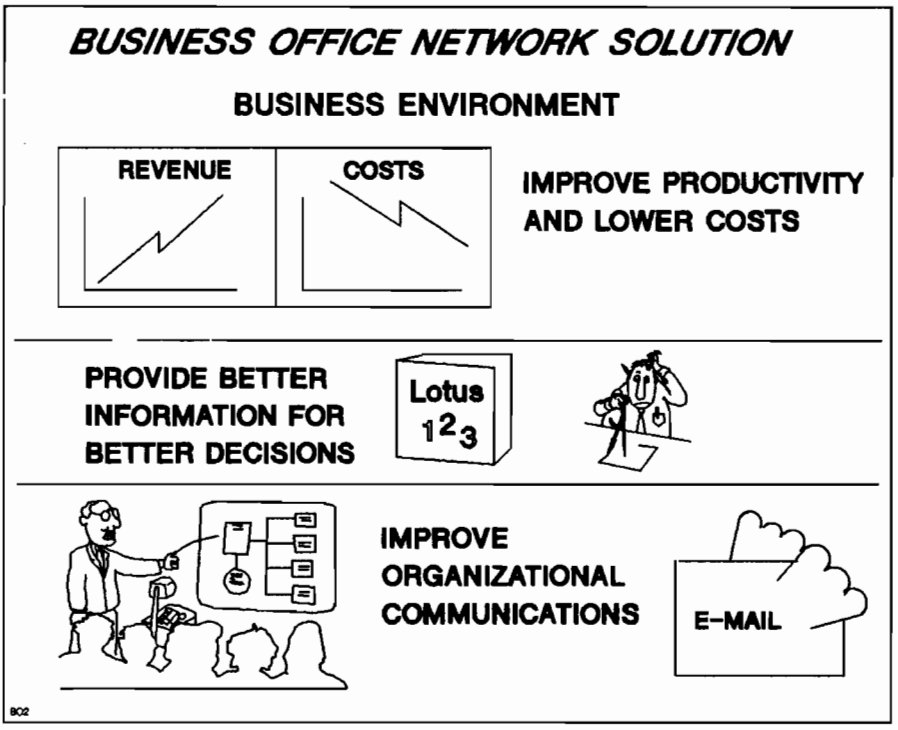


12/10/86
DB01

BUSINESS ENVIRONMENT

Most companies today recognize the potential benefits of improving the productivity of their office work force. With the advent of the PC many of these workers now have personal productivity tools like word processors, spreadsheets, and graphics packages. These are great tools for data analysis and memo creation, but, to be truly effective they need to be networked into the company's overall information and electronic communications systems to extract information for analysis and pass the results on quickly.

MIS Managers, department managers, and business managers like yourselves are looking for ways to improve productivity and get the most out of the PCs that have been purchased in your companies. You must take proactive steps to keep ahead of the competition. You know the decisions you are making today will have a tremendous effect on the future, and you may be making decisions about a new area... networking.

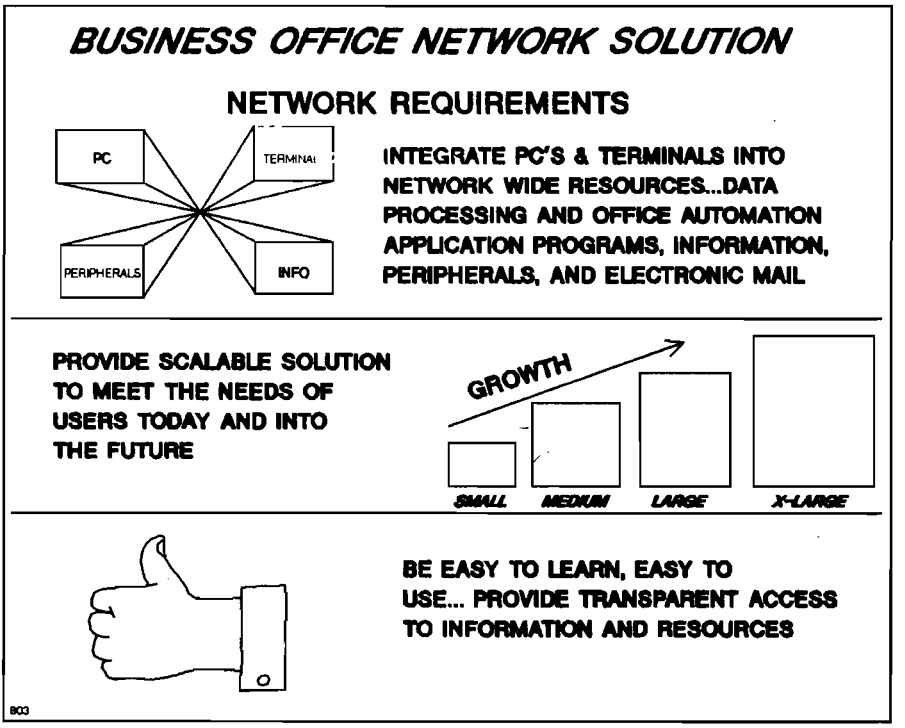


Business Office Networking Solution

NETWORK REQUIREMENTS

In order to make decisions, management must be sure that what they implement today will carry them into the future. They must be sure the vendor today will be around tomorrow, and their equipment will communicate with future equipment purchased from other vendors. This means standards based communications products or products which can readily migrate to standards.

In addition to being standards based, today's networking solutions must solve the major business problem of integrating the many workstations into the network. They must be easy to learn, and easy to use. They also must be scalable to meet the variety of sizes of business offices and be easy to grow as the company changes and grow.

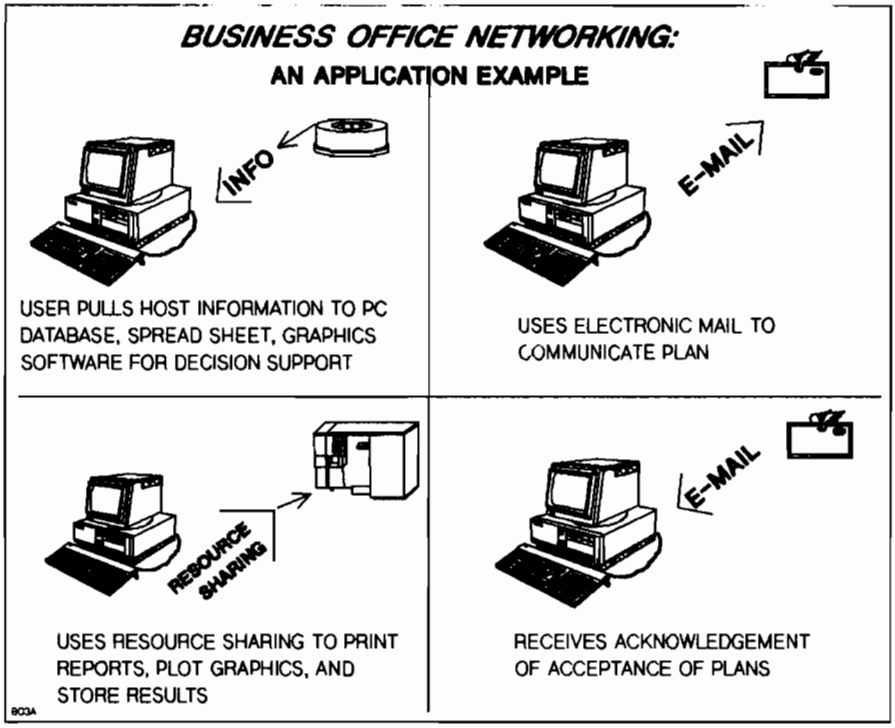


Business Office Networking Solution

AN APPLICATION EXAMPLE

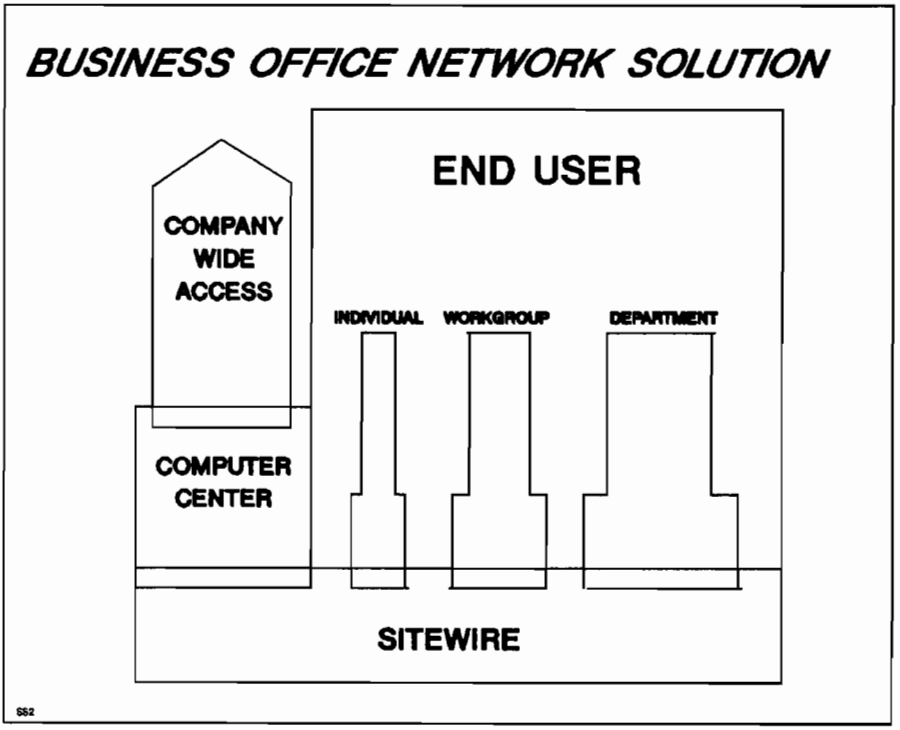
A typical example of the use of networks in the business office might be illustrated by a product manager who is trying to increase sales of a particular product (tied back to company goals of increasing sales). This Product Manager would develop a spreadsheet to analyze product sales, access company databases on sales history and advertising expenditure, download information to a PC, (directly into a spreadsheet), create graphs looking at the data in different ways and notice that there is a high correlation between product sales and local print advertising. Based on this, the product manager might create a document to report the conclusion and recommend a pilot where print advertising is increased in a key region to see if this correlation holds true. This document could be forwarded to management in the product division and to the sales and promotion people for approval and execution. This document could also be filed electronically and later accessed to plot overhead slides of the graphics for a presentation to management.

In this example, you can see how the company can better meet its business goals through powerful personal productivity tools, office tools and networking. In today's competitive environment, this type of rapid analysis and decision making can separate the winners from the also rans.



THE BUSINESS OFFICE NETWORK SOLUTION

The Business Office Network solution can be divided into four major areas or modules: Computer Center Networking, End User Networking (which is subdivided into Individuals, Workgroups and Departments of End Users), Company-Wide Access and Sitewire. These modules are designed to address specific networking problems within the business office. These basic building blocks can be combined to meet office and data processing needs. The following describes each of these modules and the alternative solutions HP can provide.



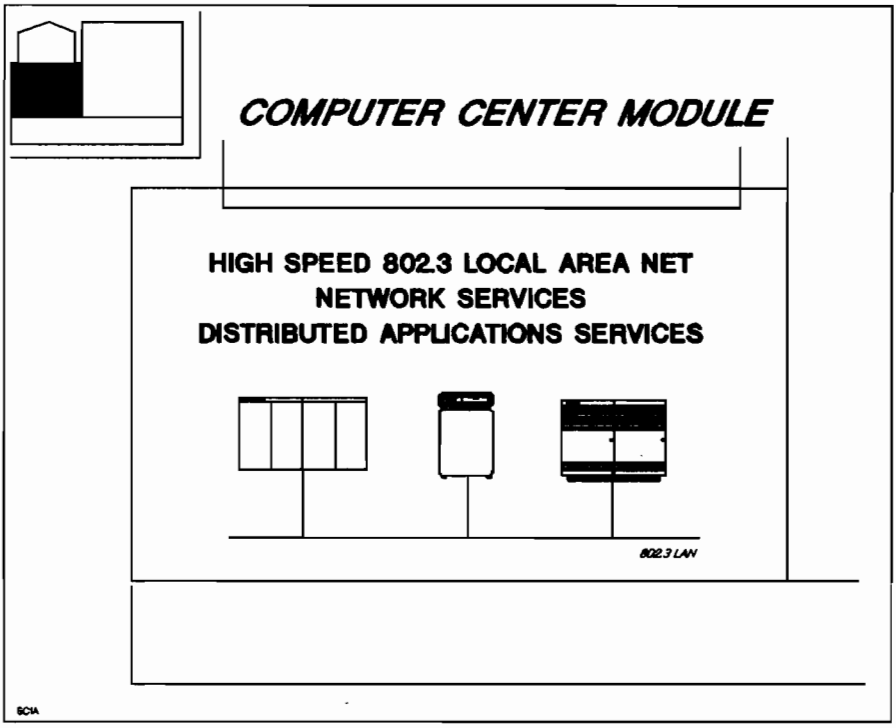
Business Office Networking Solution

THE COMPUTER CENTER MODULE

The Computer Center must provide for high speed communications between systems so applications can share peripherals and data or pass the output to other applications for input to further processing. Also, it must provide connections to the end users for terminals and personal computers to run applications, use resources and access information.

Typical applications here are financial and accounting processes, order processing, sales forecasting, parts inventory, purchasing, and electronic mail. Many involve databases which need to be shared or accessed by other applications and users.

This is best accomplished today with an 802. 3 Local Area Network and Network Services and Distributed Application Services Software. This provides capabilities such as file transfer, virtual terminal, interactive program communications, remote process management and remote database access between HP3000; in the data center.



THE END USER MODULES

End users must be able to select terminals or personal computers to meet their cost and application requirements.

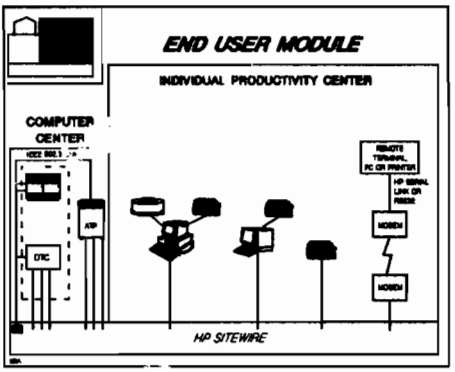
These terminals and PCs must be networkable to meet the needs of individual end users who need access to the computer center data, computing and printing resources and wants local printing capability. This individual end user may be located within the site or remotely across town, or hundreds of miles away. This can be achieved through the terminal ports on the HP3000 (ATP/DTC) and RS232 communication to the terminal or PC.

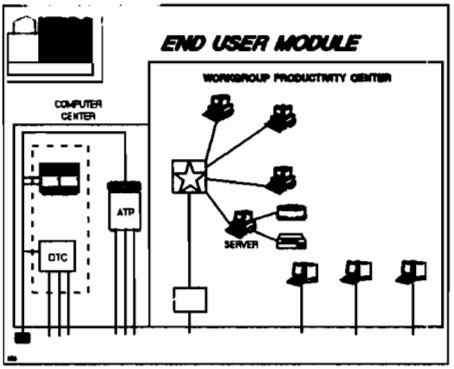
A group of end users may need to work together through their computer and network to share information and peripherals. Terminal based work groups can achieve this through terminal port connections to an HP3000 while personal computers would use a StarLan to connect to a local PC server and gain access to the HP3000.

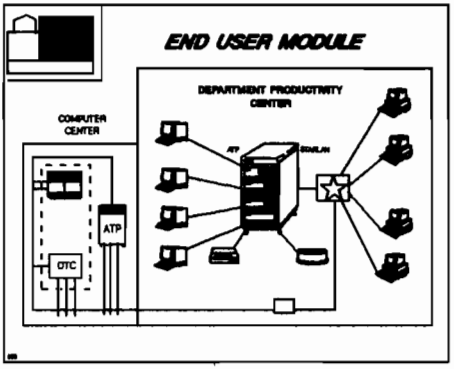
A larger department of end users who need more storage capacity, database access and data processing applications than can be provided with a PC server would choose a HP3000 as a departmental computer, networking terminals and PCs directly to it. This departmental computer would then typically be connected back to the data center through a site backbone LAN to give these users access to additional resources, databases and company- wide communications.

These individual, workgroup and departmental networking solutions can be used together to meet the overall end user networking needs, providing access to the data center, local PC servers and departmental processors. In larger business offices where multiple HP3000s are located in the data center and there is frequent access by terminal users to applications which reside on different systems, a data switch* or PBX can provide switching of terminals to different HP3000s. If occassional switching is needed, the virtual terminal capability of NS will typically meet the need.

*HP has currently not verified the operation of any data switch, however, many are installed at HP accounts today.







Business Office Networking Solution

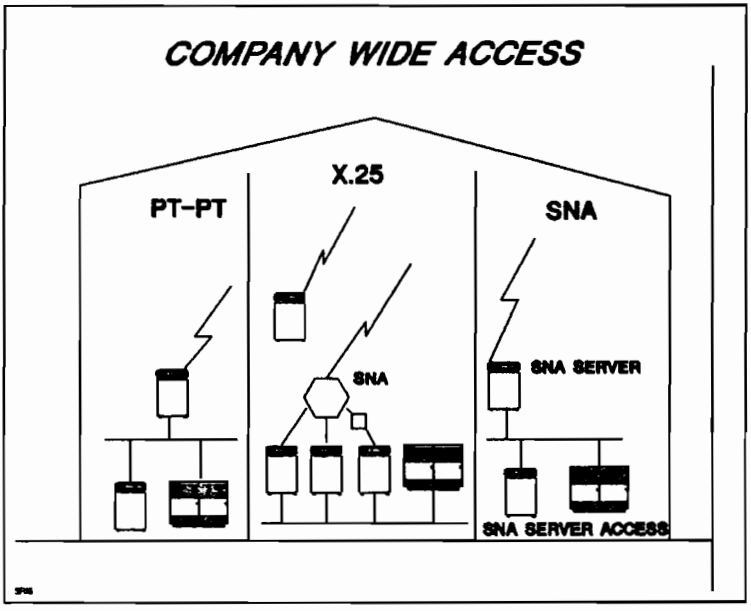
COMPANY WIDE ACCESS

The selection of the company wide access method depends largely upon the company-wide backbone network that is in place. If, for example, your company has selected X.25 for the backbone, then an X.25 access method would be required.

The selection of the company-wide access method depends largely upon the company-wide backbone network that is in place. If, for example your company has selected X.25 for the backbone, then an X.25 access method would be required.

If your company is small and has not selected a company backbone, then either point-to-point or X.25 can be used. If you are connecting 2 or 3 sites together, point-to-point is typically the best alternative. For more than 3 sites, X. 25 will typically be the best solution.

If your company has IBM systems and has selected SNA as the company-wide backbone, then the SNA link may be the lowest cost alternative. However, if multiple 3000 sites need to connect to one company headquarters site, it is often better to place an HP3000 at the headquarters site, connecting this to the IBM system using the SNA link and use this as a "gateway," connecting to the multiple 3000 sites via X.25. Creating this "subnet" within the larger SNA Network will often provide the lowest cost solution.

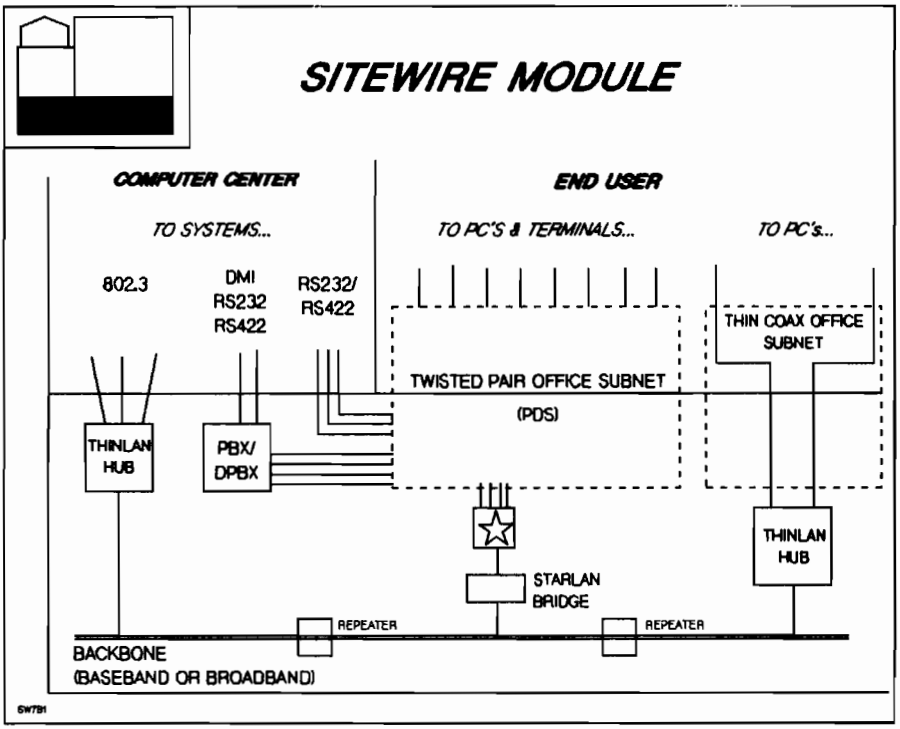


THE HP SITEWIRE MODULE

Most business office networks are located in buildings where the cost of running wire to connect end users and departmental computers together and back to the computer center is expensive. The HP Sitewire module provides alternatives to best meet the needs of a particular site and build

Depending upon the size and communication needs, HP recommends a baseband or broadband site backbone. Baseband is lower cost than broadband and meets the communication requirements of most business offices, making it the usual choice. Broadband covers very large distances and provides multiple channels for data and video, but is more costly. Many large manufacturing sites are opting for broadband and therefore offices in these facilities may use a broadband backbone.

Connecting to the backbone are office "subnets". These subnets will generally be HP Starlan running twisted pair wire. This is the lowest cost alternative for connecting workstations and often can utilize existing building wire. HP Starlan allows for easy changes and moves of the office subnet. For office subnets which connect the HP Touchscreen PC, the HP ThinLan can be used with its low cost coax cabling. These HP StarLan and HP ThinLan subnets can be combined to meet the needs of your users office configuration. The twisted pair wiring also allows connections of PC and terminals to the computers in the computer center or to PBX's for switched connections to the computers in the computer center.

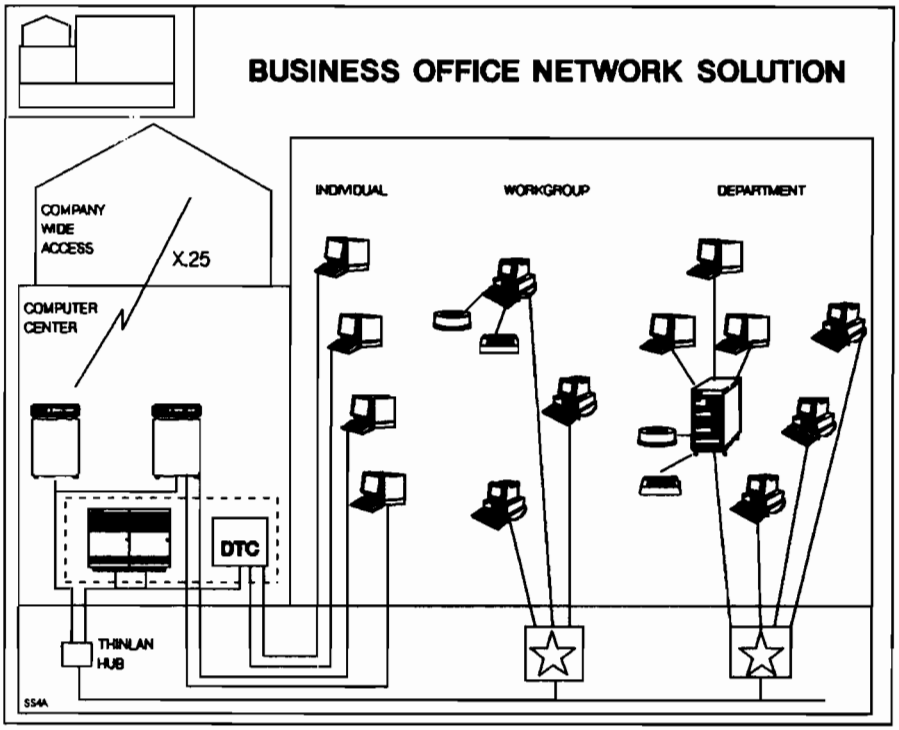


Business Office Networking Solution

PUTTING IT ALL TOGETHER-THE BUSINESS OFFICE NETWORK SOLUTION

Putting these modules all together to form an overall Office Network solution is simple and straight forward, yet provides a complete and comprehensive solution. The End User modules connect through the Sitewire module to the Computer Center. The Computer Center connects to the Company-wide Access module to connect with other sites.

HP is in a leadership position today with the low cost HP StarLan and state-of -the-art PC integration today. We offer a total solution that is scalable to meet the needs of your office today and tomorrow. HP can deliver a Business Office Network Solution today and because its based on industry standards it will be around tomorrow. HP AdvanceNet: Connecting Information with People.



Business Office Networking Solution

TROUBLE FREE UPDATES WITH AUTOINST

Kathleen A. Riley
Response Center Engineer
Hewlett Packard Response Center
3300 Scott Boulevard
Santa Clara, California 95054

ABSTRACT

In addition to death and taxes, one thing that all HP3000 system managers can be sure of is that, at some point, they will be required to update their systems to new versions of the MPE operating system. These updates are normally managed by AUTOINST, a system program which is shipped on FOS (Fundamental Operating System) tapes sent to HP3000 customer sites. If no problems are encountered during an update, the procedure can be relatively painless. If problems do occur, however, problem correction and repetition of aborted steps can be time-consuming and frustrating.

Fortunately, the majority of AUTOINST problems can be avoided by taking some preventative steps before the update is attempted. This paper explains the AUTOINST process, describes ways that users can prevent common installation errors that may occur during system updates with AUTOINST, and presents troubleshooting hints that may be used to diagnose and correct AUTOINST problems with a minimum of time and user effort.

INTRODUCTION

AUTOINST was first introduced with T-MIT and has been changed (and considerably enhanced) several times since then. It was developed to enable customers to install their own operating systems without requiring the services of an SE on site. AUTOINST manages the installation process by restoring files, streaming jobs, setting up the accounting structure and environment for installation, and running a program which creates a cold load tape with all SL and system program changes necessary to run MPE and all purchased software. The material in this paper was prepared based on UB-delta-1 versions of AUTOINST; subsequent changes and improvements have not been included.

This paper is divided into three parts. Part I is an overview of the AUTOINST process, part II is a set of recommendations for preparing for an update, and part III is a list of common AUTOINST errors and recommended recovery procedures.

I. AUTOINST OVERVIEW

The following is a brief explanation of the steps performed by AUTOINST during its execution. As AUTOINST successfully completes each step, it passes this information to a disc file named HPINSTFL. If AUTOINST aborts or is terminated for any reason, it will resume after the last successfully completed step when it is run again, based on the information in this file. Users can modify HPINSTFL using the MPE editor if it is necessary to repeat a step or jump forward in the process. HPINSTFL is created when AUTOINST is invoked; if you delete the HPINSTFL file, either accidently or intentionally, AUTOINST will build the file and start from the beginning when run again.

Please note that the steps below only include the steps performed by the AUTOINST program; they do not include the update from the FOS tape before running AUTOINST or the startup from the new cold load tape after the program has finished.

STEPS IN THE AUTOINST PROCESS

1. ENVIRONMENT IS SET

   This step is invoked each time the program is run. In this step, AUTOINST does the following:

   - reserves enough space on ldev 1 to do a cold start by building a file which takes up 17000 contiguous sectors of disc space.

   - sets the outfence to 14, the jobfence to 8, and the limits to 1,1.

   - builds the accounts used for installation if not present; groups and special capabilities are added in step 3.

2. PROGRAM AND DATA FILES FROM THE FOS ARE RESTORED

   Using an indirect file brought in with the update, AUTOINST restores the files from the FOS tape and then determines if the restore completed successfully. The restore listing is copied to a permanent file FOSLIST, which may be examined for errors. Errors during this step will result in an INSTERR #5.

3. ACCOUNTING STRUCTURE IS CREATED

   AUTOINST streams a series of jobs which create the accounting structure used in the remainder of the installation. It checks to see if each job completed successfully; if any are missing or terminate abnormally, this step fails with an INSTERR #4.

4. FILES FROM SUBSYS TAPE ARE RESTORED

   AUTOINST restores @.@.@ from the SUBSYS tape. The restore listing copied to a disc file named SUBSLIST. Errors during this step will result in INSTERR #6.

5. INSTALLATION OF NON-INSTALLABLE PRODUCTS IS CONSIDERED

AUTOINST checks to see if file sets corresponding to different special products (generally office products and HPTREND) are on the system and then prints out a message indicating the products which must be manually installed. If the user indicates that he/she would like to stop and install the products at this time, AUTOINST terminates to allow this. Before terminating, it updates the HPINSTFL file and puts a "1" in the space for "special products installed". This flag indicates to AUTOINST that the HPPL@ accounts, which hold the installation files for these products, should be purged when AUTOINST has finished its last step. For this reason, you must install all products if you choose to stop at this point. Alternatively, you may edit the HPINSTFL file to indicate that special products were not installed (replace the "1" with a "0") and the accounts will not be purged.

Note that only a small number of products must be separately installed; most (such as languages, etc.) are automatically installed by AUTOINST and are not be listed at this step. Installation instructions for the separately installed products may be found in the update manual.

6. THE JOB DUMP IS CREATED

Using another program (INSTALL), AUTOINST checks the group USL.SYS (which is built during the installation process and holds any USL files which are to be inserted into the system SL), and uses information about the files in this group and their corresponding products to build a job stream which adds or replaces segments in the system SL as appropriate. The program also creates the DUS tape for the system.

7. THE JOB DUMP IS STREAMED AND AUTOINST TERMINATES

AUTOINST streams the job DUMP from the previous step; this job creates a new cold load tape containing the new system SL and files in PUB.SYS. Because of the length of time required to add or replace SL segments, the job can run from a few minutes to over 2 hours (depending on the number of products that were purchased) before beginning to write to the tape drive. If you do have a

large number of products on your system, this is a good time to go get a cup of coffee. Purchased products (languages, office products, etc.) will only be available after this tape has been created and the system has been cold started from it.

When the job is finished and the cold load tape has been created, AUTOINST cleans up some of its utility files, sets the jobfence and outfence back to 7, issues a message to do a cold start with the new cold load tape, and terminates.

II. PREPARING FOR AN UPDATE

With proper preparation, most of the problems encountered during installations may be avoided. The following is a list of suggestions for preparing for the update which are based on my own experience using and supporting the product. Although some of the suggestions here may also be found in the update manual (they are included here for emphasis), users should follow the preparation recommendations in that document as well.

1) Check to make sure all materials are present -- update manual, FOS tape, and SUBSYS tape.

2) Always do a full backup immediately before the update.

3) Check the packing list and make sure that this includes all special products (including disc caching) that you expect to have on the tape. FOS products (IMAGE, VPLUS, KSAM, etc.) will not be listed; these are included automatically. If you have had products added to your system since the last update (i.e. the product was sent to you on a special tape) or have had a problem with missing products before, make sure that you have these on the tape. If you are missing any products, do not proceed with the installation -- contact your account SE or local office immediately.

4) Purge off all installation accounts which are used only by the installation process. This includes all the HPPL## accounts and, unless they are used at your installation, the ITF3000 and SUPPORT accounts. If you have the HPOFFICE and/or RJE accounts on your system, but they contain no files, you should purge these as well. By performing this step, you are greatly decreasing the likelihood of running into an error during the creation of the installation accounting structure.

5) Using the instructions in the update manual, make sure that the passwords and udcs on all groups, users, and accounts used in installation (HPPL##, SYS, SUPPORT, and any others listed in the manual) are removed from the system. You must also check to be sure that certain users (MGR.TELESUP, FIELD.SUPPORT, and others listed in the manual) and the associated accounts have AM capability. If you perform step #4 above (purging off the installation accounts) this task will be fairly

simple, since very few of the affected accounts will be present on the system. Note that your system will be relatively unsecured once the passwords have been removed; you may wish to disconnect your modems during the update if this is a concern.

6) Using the installation manual as a guide, remove any disc space limitations on accounts or groups affected by the update. Again, most of these accounts can be purged first so this should only affect a few accounts. Use the REPORT command to see if a group has limits on the amount of disc space it may use. To remove the disc space limitations, use the ALTACCT or ALTGROUP commands with the "FILES=" (no value given) parameter.

7) As indicated in the installation manual, your must purge HPINSTFL and FILELIST.PUB.SYS from the system (these are utility files used in AUTOINST), rename SYSSTART, and perform various other housecleaning activities. DO NOT bypass these instructions, even if you have done an installation before. The renaming of the SYSSTART file and removal of udc's in SYS is particularly important because these files often perform tasks such as allocating files, which could prevent AUTOINST from restoring new versions of the programs associated with these products.

8) As indicated in the manual, make sure that you have at least 17,000 sectors of contiguous disc space on ldev 1. If you do not, recover lost disc space and use VINIT/CONDENSE to free up disc space; if you still do not have enough disc space, purge files until enough space is obtained. Note that this step only ensures that you have enough space on ldev 1 to perform an update or cold start; it does not ensure that you have enough space to install all files on your FOS and/or SUBSYS tapes.

9) In addition to having enough disc space to perform an update or cold start, you must also have enough space on disc to restore all files on the FOS and SUBSYS tapes. The exact amount of disc space required will vary based on the number of products which are already on the system (i.e. new files will simply replace old ones), the size of the files being restored, the size of the "useful" chunks of disc space on your system, and the

products which you are bringing in on the SUBSYS tape. If you know that your system is low on disc space, you may wish to store off some big accounts before attempting the update in order to increase the likelihood of a successful update.

10) Because the installation process creates a large number of utility groups and accounts, it also uses a significant amount of directory space. The exact amount of directory space used by AUTOINST is impossible to determine because the number of additional sectors utilized depends on the existing directory structure and is a factor of some fairly complex directory algorithms that are beyond the scope of this paper. The best rule of thumb that I can provide you with is an accounting of my own experience. I have found that most installations normally require approximately 1000 sectors of directory space (as of UB-MIT) to perform an update, assuming that the installation accounts do not initially exist. If the difference between your "min" directory size and configured directory size (you can get these values in SYSDUMP) is at least this number, you probably (and I make no promises here) will not have any problems with directory space. If this is not the case, but the difference between "used" directory size and configured directory size is somewhat above this number, you may have enough "holes" in your directory structure to accomodate all the new entries. If you do not find yourself in either of these situations, the likelihood of running out of directory space increases as you stray further from these numbers. I would never recommend that anyone attempt an update without at least 600 sectors of directory space.

There are two ways to increase directory space, and neither is particularly easy. The first is to increase the configured directory space on the system, which requires a reload. If you know that you are going to be updating your system in the near future, and have insufficient directory space, the smoothest procedure all around may be to schedule a reload. Keep in mind that, in order to change the size of the directory, you must create a sysdump tape with the increased directory size and reload from that tape (i.e., you will not be prompted for directory size changes if you reply "Y" to changes when the system is coming up).

The other way to free up directory space is to purge off some big accounts (preferably with lots of groups) from the system. Before doing so, you may want to run buldacct (a utility commonly found in telesup) and generate some job streams to rebuild those accounts after the update is completed. You may alternatively use the "CREATE=" options in RESTORE to rebuild the accounts, but they will then be created with default access and capabilities. Purging off files without purging off the account structure will also free up directory space, but files take up much less room in the directory than account or group structures and you will have to purge off a large number of files to have much effect on directory size.

11) Have a fairly new tape handy to create the new cold load tape. This tape should be used a few times to make sure it is good. If you create a bad cold load tape, you may have to repeat the entire update process.

12) If you have been having hardware problems with your tape drive or system, do not proceed with your update until these problems have been resolved and you are fairly certain that you have a "stable" system. Updating your operating system is a fairly tape drive intensive process. Clean the heads on the tape drive before the update and before creating the cold load tape and make sure that you are up to date on all maintenance on your tape drive and system.

III. COMMON AUTOINST ERRORS AND RECOMMENDED RECOVERY

The following is a list of some of the more commonly encountered AUTOINST errors and steps that may be taken to recover from them.

INSTERR #2 - Not enough disc space on ldev 1.

Whenever AUTOINST is invoked, it reserves enough space on ldev 1 to perform a successful cold start from tape. This error indicates that it could not find this space. You must free up 17000 sectors of contiguous disc space to continue. As explained in the update manual, you may recover the space you need by

doing a cool start with the recover lost disc space option (do NOT attempt an update or cold start) if you think you may have lost disc space, or by storing to tape and then purging files until you have enough space on ldev 1. You may then use :VINIT with the COND 1 option to get the free space into a contiguous block.

INSTERR #4 - Creation of the accounting structure has failed.

This is the most frequently encountered AUTOINST error, because its successful completion depends on a number of factors. To determine the cause of the failure you may do the following:

1) Compare the completion messages which have been issued to the console with the list of messages in the AUTOINST documentation. Any missing completion messages indicate jobs which did not terminate normally.

2) Use the logon/logoff console messages to find the job numbers corresponding to any jobs which did not complete, and use SPOOK5 to examine the $STDLISTs for these jobs and determine which error(s) caused the job(s) to abort. Do not worry about warning messages or errors in statements which were preceded by continue statements. If you cannot find the cause of the error in the job listings which you are examining, look at all the job streams.

   NOTE: If you are installing T-delta-4 or earlier versions of MPE, you may run into this error if the output device for the streams device (customarily ldev 10) is an ldev instead of a device class. If this is the case, and you are certain that all the job streams completed successfully, you may edit the file HPINSTFL to indicate completion of this step and continue.

3) Based on the error(s) which caused the job(s) to abort, fix the problem and continue. The following is a list of common errors and their corrective actions.

   - "MISSING PASSWORD. (CIERR 1444)" -- you left a password on the account, user, or group for the job which is being streamed by the job being

examined. Remove the offending password(s) and run AUTOINST again.

- "ACCOUNT EXISTS, USER NAME DOESN'T. (CIERR 1438)" -- The account existed previous to the update, but the user which it expects to find as account manager is missing. Create the user in the account and make sure to give it AM, IA, BA, ND, and SF capabilities. Run AUTOINST again.

- "THIS COMMAND REQUIRES ACCOUNT MANAGER (AM) OR SYSTEM MANAGER (SM) CAPABILITY. (CIERR 958)" -- the account existed previous to the update, but the user which it expected to find as account manager does not have AM capability. Sign on as manager of the account in question and do an ALTUSER on the signon user in the job and give it AM, IA, BA, ND, and SF capabilities. Run AUTOINST again.

- "DIRECTORY OUT OF SPACE -- SYSTEM PROBLEM. (CIERR 915)" -- The system cannot create any more directory entries for the group/account it is trying to build. This is a serious problem because increasing the amount of directory space may require a reload. See part 10 of the section "PREPARING FOR AN UPDATE" for suggestions on increasing the amount of directory space available. If you choose to stop the update at this point and complete it after you have corrected the problem, you may go back to the old operating system by following the directions under "RETURNING TO YOUR FORMER OPERATING SYSTEM" at the end of this section.

INSTERR #5 - FILES FROM FOS TAPE NOT SUCCESSFULLY RESTORED

The system was not able to restore all the files it expected from the FOS tape. Go into EDITOR, text in the file FOSLIST.PUB.SYS, and find the files that were not restored (hint: give the command 'C "NOT" TO "NOT" IN ALL' to get a list of only those files that were not restored). The problem may be one of the following:

- "I/O ERROR READING UNLABELED TAPE" and other transmission errors -- clean the tape heads, check and tighten the cabling to your disc drives, and run AUTOINST again to repeat this step. If only one or

two files are missing, you may wish to simply attempt to restore the files manually, then adjust the current status in HPINSTFL to indicate that the step completed successfully. If the problem appears to be a hardware error on your tape drive, you should correct the problem before continuing with the installation.

If the restore continues to abort on the same file, there may be a problem on your tape. You can usually determine from the file name what the file is and whether lacking that file will affect the installation or your ability to use the machine after the update is completed (e.g. missing COUR88A.HPENV.SYS will certainly have far less impact than missing QUERY.PUB.SYS, unless you are a text processing shop with heavy 2688 usage). If you choose to go on without the file, you can edit the current status in HPINSTFL and run AUTOINST again to continue at the next step, then contact your account SE or local HP sales office about getting a good copy of the file. If you choose to give up at this point, restore the system back to its original state before the update (see "RETURNING TO YOUR FORMER OPERATING SYSTEM" at the end of this section) and contact your local office about getting a new tape.

- "OUT OF ACCOUNT DISC SPACE" or "OUT OF GROUP DISC SPACE" -- The account existed previous to the update with a limit on the amount of disc space it may occupy. Sign on as MANAGER.SYS and do an ALTACCT with the parameter ";FILES=" to remove space limitations on the account, and then sign on as the account manager and do an ALTGROUP on each group in the account with the same parameter. You may then run AUTOINST again to redo this step and continue.

- "OUT OF DISC SPACE" -- You do not have enough disc space on the system to restore the file(s). Purge off files from accounts not related to the update (assuming you have a full backup of the system) until you have plenty of disc space, and then run AUTOINST again. Again, you may wish to manually restore the files and edit HPINSTFL if only one or two files were missing, or choose not to restore the files at all if they are unnecessary for your installation (e.g. file type HPENV at an installation with no laser printers).

- "NOT ON TAPE" -- This indicates that the FOS tape does not match the indirect file which was restored

when the system was updated. In other words, you did not update from the same FOS tape that you have on your tape drive (or you forgot the update step completely). If you simply have the wrong tape on the drive, put the correct tape on and run AUTOINST to do this step again. If you did not update before running AUTOINST, or if you used the wrong FOS tape, purge HPINSTFL.PUB.SYS, update from the correct FOS tape, and run AUTOINST to start the process again.

- "FILE IS LOADED" -- The old version of this file has been allocated and cannot be overwritten. Use the DEALLOCATE command to deallocate the file(s) and run AUTOINST again to repeat this step and continue.

INSTERR #6 -- SUBSYS TAPE WAS NOT SUCCESSFULLY RESTORED

The system was not able to restore all the files on the SUBSYS tape. Go into EDITOR, text in the file SUBSLIST.PUB.SYS, and find the files that were not restored (hint: give the editor command 'C "NOT" TO "NOT" IN ALL' to get a list of only those files that were not restored). The problems and remedies are the same as those for insterr #5, above; the only difference is that all files on the SUBSYS tape will be necessary for the successful installation of your subsys products; if you absolutely cannot get a file off the tape, you will have to determine if you can live without the corresponding product once the system is up and continue or retreat as appropriate.

INSTERR #8 -- COLD LOAD TAPE NOT SUCCESSFULLY CREATED

Use SPOOK5 to examine the spoolfile for the job "DUMPJOB" (you should have seen its logon message at the console). The problem is normally one of the following:

- "ERROR WRITING TO TAPE" (or similar i/o errors) -- clean the tape heads again, put a different, known good tape on the tape drive, and run AUTOINST again.

- TOO MANY CODE SEGMENTS" -- on T-delta-4, T-delta-5, T-delta-6, UA-MIT, and U-MIT only -- You have so many products on your SUBSYS tape that all the required segments do not fit in your system SL (this problem was eliminated when the SL

size was increased with UB-MIT). This problem is seen when using a "complete" (i.e. every available product) subsys tape with the above MIT's. You must either go back to your old MIT and update at a later time from a customized subsys tape, or edit the job stream "DUMP" so that it does not add so many entries to the system SL. This latter procedure must only be done if you know what you are doing and is, of course, unsupported by HP (as is updating from a "complete" subsys tape).

- "SEGMENT ALREADY DEFINED, SEGMENTER ERROR 0" OR "DUPLICATE ENTRY POINT" -- Not only are you using a "complete" subsys tape for the installation, but you neglected to install one of the versions of HPSPELL (so it attempted to install both versions). The situation is salvageable if you edit the job stream "DUMP" to remove all lines referring to files "U00D561A" and "U00U561A" if you wish to install American HPSPELL and all lines referring to file "U00U561B" if you wish to install "dual" (American/British) HPSPELL. You may then run AUTOINST again to re-do the cold load creation step.

OTHER ERRORS/PROBLEMS

- Wrong tape mounted for subsys restore -- If any tape is mounted when AUTOINST checks the tape drive to restore from the subsys tape, AUTOINST will automatically start reading the files from that tape. If you break and abort AUTOINST before it has finished reading the tape, you may simply mount the correct tape and re-run AUTOINST to install the correct files. If the restore completes before you catch the error, you should abort AUTOINST, text the HPINSTFL file into the MPE editor, change the "current status" under "progress check point" to step 3 (accounting structure has been created), and run AUTOINST again.

- "NONEXISTENT PERMANENT FILE STORCHK" -- This error occurs during the FOS or SUBSYS restore and indicates that AUTOINST was unable to save a temporary file on disc in the group PUB.SYS. This usually indicates that SAVE access has been taken from PUB.SYS or the group is out of file space. Use LISTDIR5 to determine the cause of problem, correct it, and rerun AUTOINST.

RETURNING TO YOUR FORMER OPERATING SYSTEM

For whatever reason, you may find it necessary to discontinue the update and return to your former operating system. The following guidelines should help you accomplish this.

- If you were updating from a pre-V/E system (Q-delta-2, V/P, etc.) and ran PCONVERT before beginning the update, see the note at the end of this section before attempting a cold start.

- If you discontinue the update before the FOS restore, cold start from your last cold load tape (or the first tape of your most recent backup). Using the system update manual (see the section entitled "BRING THE SYSTEM UP"), restore the system back to its original state (passwords, udc's, etc.).

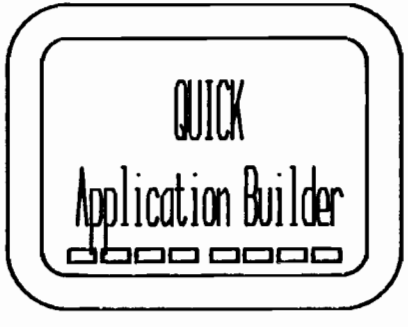
- If you discontinue after the FOS restore but before the SUBSYS restore, perform the above step and then restore @.@.SYS, @.@.TELESUP, and @.@.SUPPORT from the last full backup. You many also purge off the HPPL## accounts.

- If you discontinue after the SUBSYS restore, follow the same procedure as you would have after just a FOS restore (above), and additionally restore any files that were written over in the SUBSYS tape restore. Check the SUBSLIST to see which files were restored; if applicable, restore @.@.HPOFFICE, @.@.ITF3000, and/or @.@.RJE. Do not worry about any files in the HPPL## account or in USL.SYS.

- SPECIAL NOTE FOR THOSE RETURNING TO PRE-V/E SYSTEMS: If you were updating from a pre-V/E system and ran PCONVERT before the update, run PCONVERT again to convert your tables back to the pre-V/E table structure. Cold start from your last cold load tape (or the first tape of your most recent backup) and continue as above. DO NOT RUN PCONVERT IF YOU DID NOT HAVE TO RUN IT TO BEGIN THE UPDATE PROCESS. IF YOU RUN PCONVERT AND ARE NOT GOING BACK TO A PRE-V/E SYSTEM, YOU WILL FORCE YOURSELF INTO A RELOAD.

QUICK: Procedures "The Ins & Outs"

David G. Robinson
ROBINSON, MARRAGE & COMPANY
11693 San Vicente Blvd
Suite 168
Los Angeles, CA 90049



ABSTRACT

PowerHouse: QUICK Procedures "The INs and OUTs"

The latest release of POWERHOUSE, 5.01, contained the inclusion of 9 additional procedures making a total of 22 procedures available to the screen designer. Procedures are the foundation of all QUICK Screen's Logic and act as exit points for the designer to initiate standard or customized processing.

What are these procedures ? How do they work ? Where do I use them ? These are some of the basic questions that need to be addressed before any designer begins implementing their system of QUICK Screens.

This paper describes the various modes that the procedures operate in. The implementing and timing of these procedures, along with the many procedural verbs, predefined conditions and items that play an integral part of processing in QUICK Screens.

What are PROCEDURES ?

- High-level USER exits
- Points of Intervention in QUICK's processing
- Default Procedures
  - ENTRY & APPEND
  - PATH & FIND & DETAIL FIND
  - DELETE & DETAIL DELETE
  - UPDATE
- QDESIGN ---> GENERATES Default Procedures
  - Designer statements
- BUILD List
  - Displays Procedures

There are 14 additional Procedures !

- Customization
- Additional Edits
- Security

QUICK
APPL. BUILDER

Procedural Statement

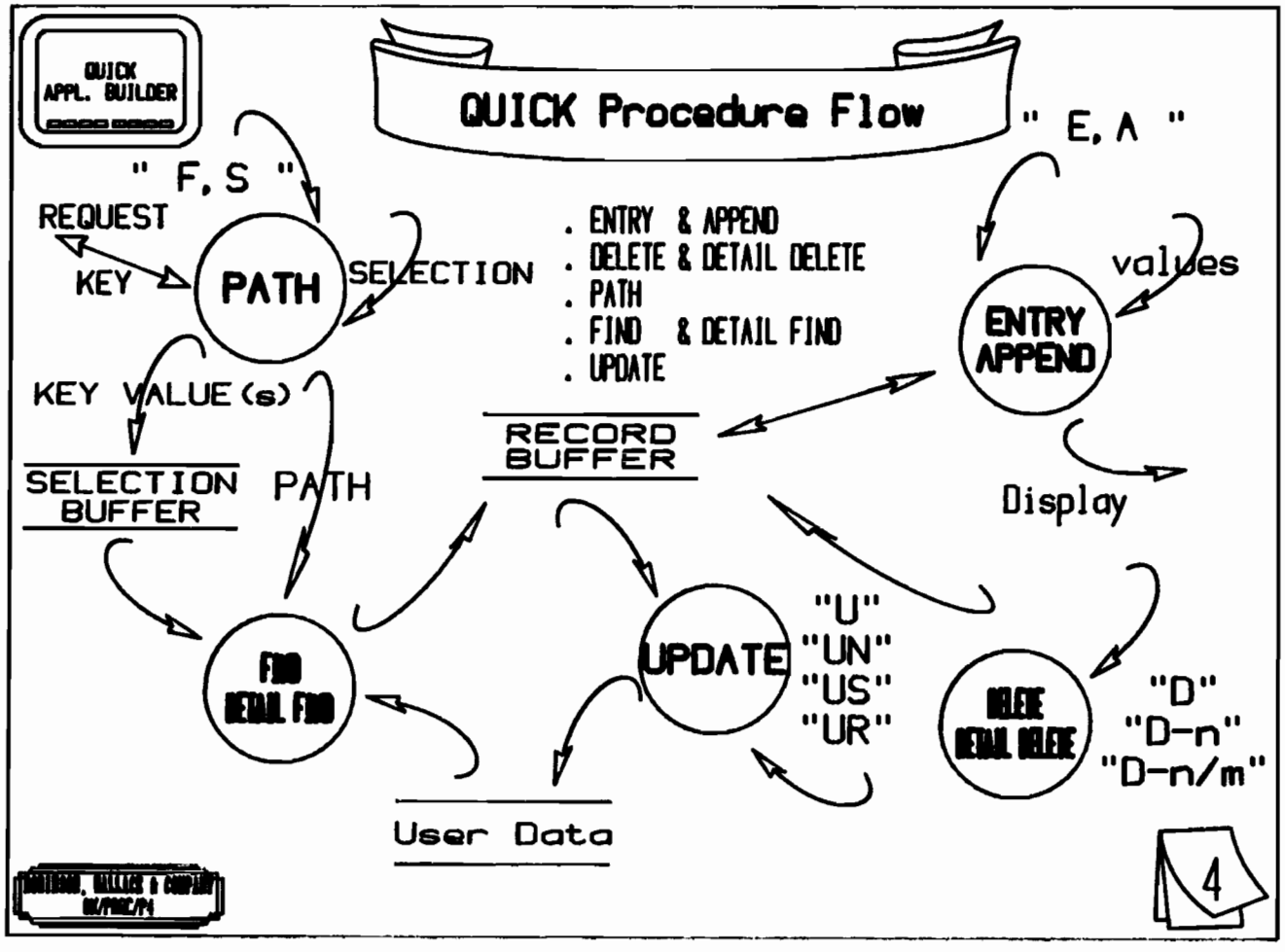
Basic Syntax:

. Where? End of SCREEN design: just before BUILD stmt

. How? Start with PROCEDURE statement
PROCEDURE type (name)
> PROCEDURE PROCESS qty-requested

. Next: Procedure body can consist of the following:
> BEGIN
. One or more Procedural statements
. Verb / ACCEPT, LET, GET, PUT
. Conditional statement / IF ... THEN
. Repetitive statement / FOR
> END

```
> PROCEDURE PROCESS qty-requested
> BEGIN
>   LET commission = qty-requested * price * com-rate
>   IF  qty-requested  GE 100
>       THEN LET commission = commission * 1.05  ; 5% bonus
> END
```


QUICK
APPL. BUILDER
QUICK Procedure Flow
" F, S "
" E, A "
REQUEST
KEY
PATH
SELECTION
. ENTRY & APPEND
. DELETE & DETAIL DELETE
. PATH
. FIND & DETAIL FIND
. UPDATE
values
ENTRY
APPEND
KEY VALUE(s)
SELECTION
BUFFER
PATH
RECORD
BUFFER
Display
FIND
DETAIL FIND
UPDATE
"U"
"UN"
"US"
"UR"
DELETE
DETAIL DELETE
"D"
"D-n"
"D-n/m"
User Data
4

QUICK
APPL. BUILDER

PROCEDURE ENTRY

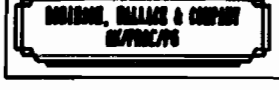
- Allows data entry of new records
  - Except ---> Repeating PRIMARY Records ---> APPEND
    DETAIL Records ---->
- Specifies FIELDs into which data is entered & displayed
- Can override options on designer FIELD stmts
- Along with APPEND procedure, primary user of:
  - ACCEPT, EDIT, and DISPLAY verbs
- Executed when USER enters "E" in screen ACTION ---> "E"
  - Calls APPEND procedure
- Nullify by omitting from Screen ACTIVITIES option

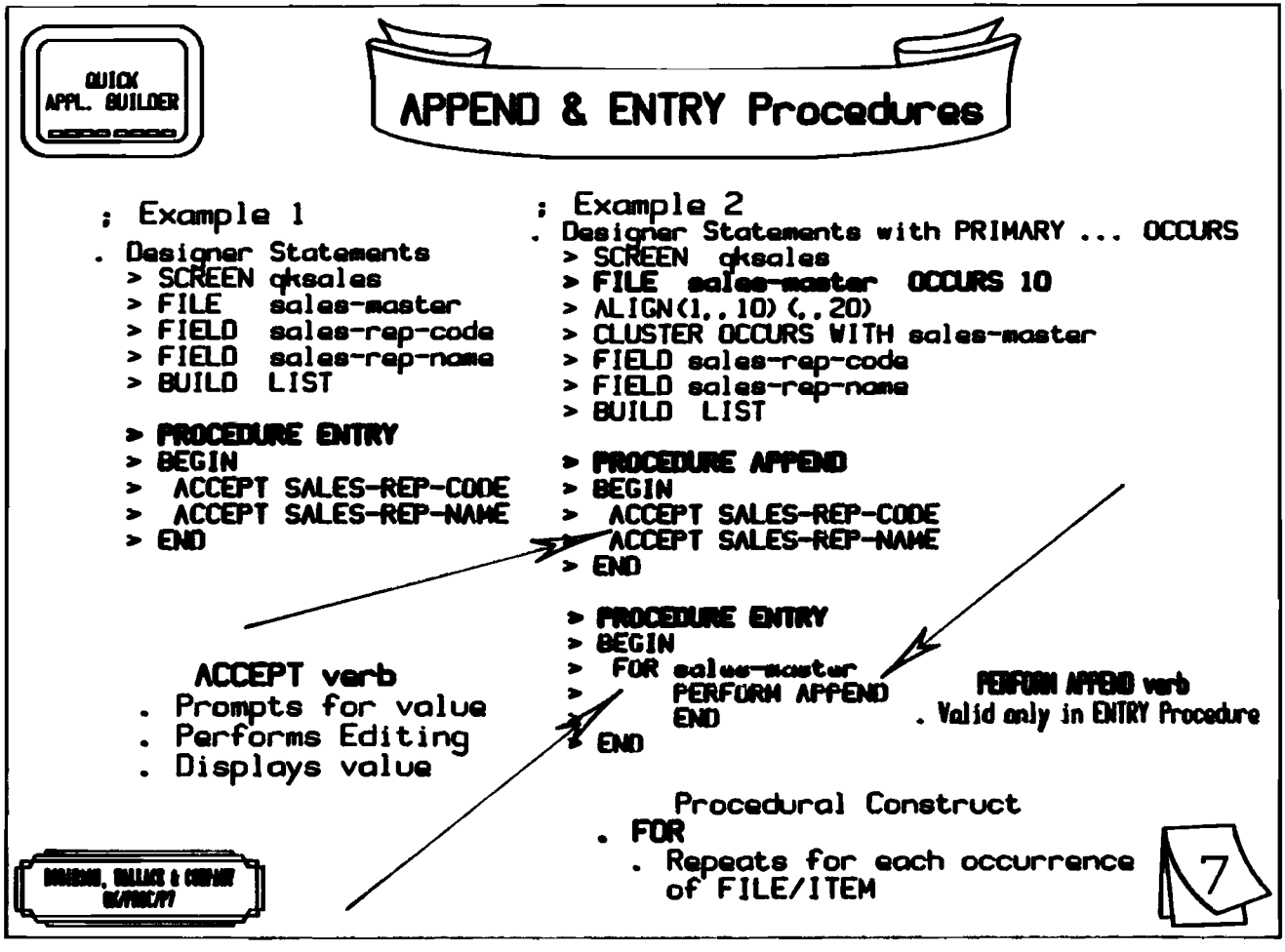
```
> PROCEDURE ENTRY
> BEGIN
>   ACCEPT  CUSTOMER-ID    OF ORDER-HDR-DETAIL
>   ACCEPT  ORDER-NO       OF ORDER-HDR-DETAIL
>   ...
>   FOR ORDER-LNE-DETAIL
>       BEGIN
>         PERFORM APPEND
>       END
> END
```

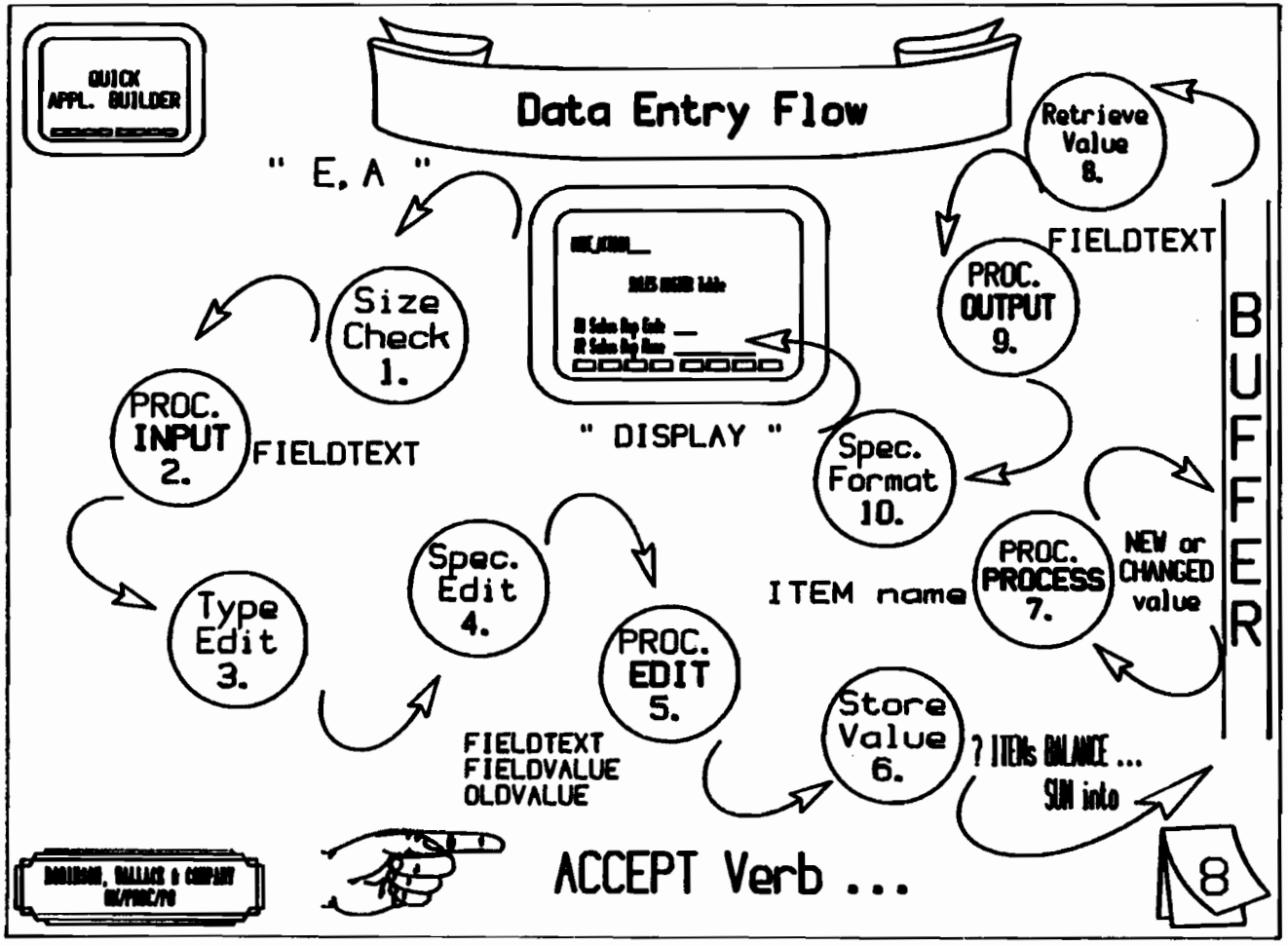
PROCEDURE APPEND

- Implements Data Entry of :
  - PRIMARY Records; if PRIMARY File OCCURS n
  - DETAIL Records; Append Mode Processing
- Executed by QUICK when:
  - Called from ENTRY procedure
  - Screen is in FIND MODE
    - USER enters "A" after subsequent update "U"
- Nullify by
  - Specifying NOAPPEND on repeating PRIMARY File
  - Omitting ENTRY and CHANGE from SCREEN ACTIVITIES
  - Using DISABLE

```
> PROCEDURE APPEND
> BEGIN
>   ACCEPT  PART-NUMBER      OF ORDER-LNE-DETAIL
>   DISPLAY DESCRIPTION      OF PARTS-MASTER
>   ACCEPT  QTY-REQUESTED    OF ORDER-LNE-DETAIL
>   DISPLAY LNE-TOTAL
> END
```

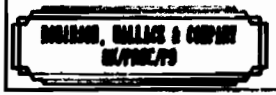



QUICK
APPL. BUILDER
APPEND & ENTRY Procedures
; Example 1
. Designer Statements
> SCREEN qksales
> FILE sales-master
> FIELD sales-rep-code
> FIELD sales-rep-name
> BUILD LIST
> PROCEDURE ENTRY
> BEGIN
> ACCEPT SALES-REP-CODE
> ACCEPT SALES-REP-NAME
> END
; Example 2
. Designer Statements with PRIMARY ... OCCURS
> SCREEN qksales
> FILE sales-master OCCURS 10
> ALIGN(1,,10) (,,20)
> CLUSTER OCCURS WITH sales-master
> FIELD sales-rep-code
> FIELD sales-rep-name
> BUILD LIST
> PROCEDURE APPEND
> BEGIN
> ACCEPT SALES-REP-CODE
> ACCEPT SALES-REP-NAME
> END
> PROCEDURE ENTRY
> BEGIN
> FOR sales-master
> PERFORM APPEND
> END
> END
ACCEPT verb
. Prompts for value
. Performs Editing
. Displays value
PERFORM APPEND verb
. Valid only in ENTRY Procedure
Procedural Construct
. FOR
. Repeats for each occurrence of FILE/ITEM
7


QUICK APPL. BUILDER
Data Entry Flow
" E, A "
Size Check 1.
PROC. INPUT 2.
FIELDTEXT
Type Edit 3.
Spec. Edit 4.
PROC. EDIT 5.
FIELDTEXT
FIELDVALUE
OLDVALUE
Store Value 6.
? ITEMs BALANCE ...
SUM into
" DISPLAY "
Spec. Format 10.
ITEM name
PROC. PROCESS 7.
NEW or CHANGED value
Retrieve Value 8.
FIELDTEXT
PROC. OUTPUT 9.
BUFFER
ACCEPT Verb ...
8

PROCEDURE Verbs & Predefined Conditions

- **ACCEPT verb**
  - Prompts for, edits and displays FIELD value
  - Performs all 10 steps of Data Entry Flow
- **EDIT verb**
  - Edits FIELD value
  - Performs steps 4, 5, and 7 in Data Entry Flow
    - Retrieves value from RECORD BUFFER and place into
      - FIELDTEXT or FIELDVALUE
  - Used in conjunction with SILENT FIELDs
- **DISPLAY verb**
  - Displays FIELD value
  - Performs steps 8, 9 and 10 in Data Entry Flow
  - Also performed when
    - Predisplay of initialized values
    - After FIND and DETAIL FIND Procedures
    - FIELDs with REFRESH option
- **PERFORM APPEND**
  - Execute APPEND Procedure
  - Valid only in ENTRY Procedure



QUICK APPL. BUILDER

PROCEDURE DELETE

- Flag RECORD for deletion
  - Deleted physically via UPDATE procedure
- Executed when USER enters ---> "D, D-n, D-n/m" in ACTION
- Only use "FOR" construct when:
  - PRIMARY File OCCURS n
  - DETAIL or SECONDARY Files OCCURS n
- Primary user of verb "DELETE"
- Nullify by omitting DELETE from screen ACTIVITIES option

```
> PROCEDURE DELETE
> BEGIN
>   DELETE ORDER-HDR-DETAIL
>   FOR ORDER-LNE-DETAIL
>       BEGIN
>         DELETE ORDER-LNE-DETAIL
>       END
> END
```

QUICK
APPL. BUILDER

PROCEDURE DETAIL DELETE

- Flags DETAIL Records for deletion
  - Physically deleted in UPDATE procedure
- Executed when USER enters ---> "D, D-n, D-n/m" in ACTION
- Nullify by omitting "DELETE" from screen ACTIVITIES option

```
> PROCEDURE DETAIL DELETE
> BEGIN
>   DELETE ORDER-LNE-DETAIL
> END
```

QUICK
APPL. BUILDER

DETAIL DELETE & DELETE Procedures

```
. Designer Statements
  > SCREEN qkorder
  > FILE   order-hdr-detail PRIMARY
  > FILE   order-lne-detail DETAIL   OCCURS 10
  > ...
  > BUILD  LIST

  > PROCEDURE DELETE
  > BEGIN
  >  DELETE order-hdr-detail
  >  FOR    order-lne-detail
  >    BEGIN
  >      DELETE  order-lne-detail
  >    END
  > END

  > PROCEDURE DETAIL DELETE
  > BEGIN
  >   DELETE order-lne-detail
  > END
```

DELETE verb

. Flags RECORD for Deletion

CORRECTION Phase: Prevents RECORD being Added
CHANGE Phase: Flags RECORD for DELETION

QUICK APPL. BUILDER

PROCEDURE Verbs & Predefined Conditions

- **DELETE verb**
  - Mark a record for deletion
  - Requires a subsequent "U"pdate to physically delete the record from the File
  - Once a record is marked for deletion, the data from that record is no longer accessible

- **FOR construct**
  - Specifies a repetitive statement
  - Options are:
    - > FOR filename/item
      - Repeat once for each occurrence of the File or item
    - > FOR n
      - Repeat for the number of times specified (255 maximum)
- **OCCURRENCE predefined condition**
  - Index of current File or Item in FOR loop
  - = 1, outside of loops
  - Use to select a particular OCCURRENCE
  - Item subscripting not allowed

```
> PROCEDURE ...
> BEGIN
>    FOR ORDER-LNE-DETAIL
>          IF OCCURRENCE EQ 5
>          THEN ...
> END
```

QUICK
APPL. BUILDER

PROCEDURE PATH

- REQUESTs KEY value(s) of PRIMARY File from USER
- Sets value of predefined item PATH, based on USER response
  - PATH=0 at beginning of PATH procedure
- Executed when USER enters ---> "F,S" in ACTION
- Primary user of verb REQUEST
  - Predefined condition PROMPTOK is true if value was entered
- FIND procedure uses PATH to determine Access Method
- Omit "FIND" from screen ACTIVITIES option

```
> PROCEDURE PATH
> BEGIN
>   REQUEST CUSTOMER-ID   OF ORDER-HDR-DETAIL
>   IF PROMPTOK
>      THEN LET PATH = 1
>   IF PATH = 0
>      THEN BEGIN
>             REQUEST ORDER-NO  OF ORDER-HDR-DETAIL
>             IF PROMPTOK
>                THEN LET PATH = 2
>           END
>   IF PATH = 0
>      THEN BEGIN
>           LET PATH = 3
>...
```

QUICK
APPL. BUILDER

PROCEDURE FIND

- Allows retrieval of Records
- Uses predefined item PATH to determine retrieval strategy
  - KEY or Sequential Access
- IF FILE with OCCURS n
  - Tries to retrieve requested number of Records
- Primary user of verb GET
  - Retrieval of Records is assumed Required
  - OPTIONAL is not valid for PRIMARY Files
- Executed when USER enters ---> "F,S" in ACTION
- Nullify by omitting "FIND" in screen ACTIVITIES option

```
> PROCEDURE FIND
> BEGIN
>    IF PATH = 1
>       THEN GET ORDER-HDR-DETAIL  VIA CUSTOMER-ID
>    IF PATH = 2
>       THEN GET ORDER-HDR-DETAIL  VIA ORDER-NO
>    IF PATH = 3
>       THEN GET ORDER-HDR-DETAIL  SEQUENTIAL
> END
```

QUICK
APPL. BUILDER

PROCEDURE DETAIL FIND

- Allows retrieval of records from DETAIL File
  - Also Files that OCCURS with DETAIL File
- Executed ---> After FIND and POSTFIND procedures
- Records retrieved according to Design Linkages
  - Linkage by KEY is required
  - Retrieval is assumed required unless OPTIONAL specified
- Like FIND procedure uses verb GET to retrieve Records
- Nullify by omitting "FIND" in screen ACTIVITIES option

```
> PROCEDURE DETAIL FIND
> BEGIN
>   FOR ORDER-LNE-DETAIL
>       BEGIN
>         GET ORDER-LNE-DETAIL OPTIONAL
>       END
> END
```

FIND & DETAIL FIND Procedures

```
;Example 1
. Designer Statements
  > SCREEN qkscales
  > FILE   sales-master
  > FIELD  sales-rep-code
  > FIELD  sales-rep-name
  > BUILD  LIST
```

```
> PROCEDURE FIND
> BEGIN
>    IF PATH = 1
>       THEN GET SALES-MASTER VIA SALES-REP-CODE
>    IF PATH = 2
>       THEN GET SALES-MASTER SEQUENTIAL
> END
```

;Example 2

```
. Designer Statements
> SCREEN qkorder
> FILE   order-hdr-detail  PRIMARY
> FILE   order-lne-detail  DETAIL  OCCURS 10
> ...
```

File Relationship MUST Exist ...

```
> PROCEDURE FIND
> BEGIN
>    IF PATH = 1
>       THEN GET ORDER-HDR-DETAIL VIA CUSTOMER-ID
>    IF PATH = 2
>       THEN GET ORDER-HDR-DETAIL VIA ORDER-NO
>    IF PATH = 3
>       THEN GET ORDER-HDR-DETAIL SEQUENTIAL
> END
> PROCEDURE DETAIL FIND
> BEGIN
>    FOR ORDER-LNE-DETAIL
>    BEGIN
>       GET ORDER-LNE-DETAIL OPTIONAL
>    END
> END
```

GET verb

. Retrieves a RECORD

PROCEDURE Verbs & Predefined Conditions

- **PATH predefined ITEM**
  - Holds an integer value that is set by PATH Procedure
  - PATH is set to zeros at start of ENTRY and FIND MODE
  - FIND Procedure interrogates to determine record retrieval
- **REQUEST verb**
  - Prompts for value required for record retrieval
  - Recommend used only in PATH and POSTPATH Procedures
  - Value is stored in SELECTION BUFFER
- **PROMPTOK predefined condition**
  - True if operator entered a value in response to:
    - REQUEST or PROMPT or ACCEPT verbs
  - True if operator entered value in FIELDTEXT by an INPUT Procedure
- **GET verb**
  - Performs actual record retrieval
  - Options similar to ACCESS statement:
    - BACKWARDS, GENERIC, NOGENERIC, SEQUENTIAL, UNIQUE, USING expressing, or VIA key
    - OPTIONAL
      - If retrieval attempt fails, GET without OPTIONAL fails with an ERROR
  - Used in FIND Procedure to obtain records from PRIMARY, SECONDARY, DETAIL and REFERENCE Files
  - Used in DETAIL FIND Procedure to retrieve records from DETAIL Files
- **ACCESSOK predefined condition**
  - Used with GET .... OPTIONAL ; true if record was retrieved successfully

```
> GET ....
> IF NOT ACCESSOK
>    THEN ERROR "..."
```

QUICK
APPL. BUILDER

PROCEDURE UPDATE

- Functions:
  - Add new records
  - Replace changed records
  - Remove records marked for deletion
- IMAGE DBMS
  - Sequences outputs to match Data Base restrictions
    - Adds master before details
    - Delete details before master
- Primary user of verb PUT
  - Usage of PUT verb outside of UPDATE ---> BACKOUT procedure ?
- Avoid ERROR, SEVERE verbs, as they will halt Procedure
  - Do edits in PREUPDATE Procedure
- Executed when USER enters ----> "U, US, US, OR UR" in ACTION
- Updates are rolled back upon any ERRORs encountered
  - Rollback Processing
- UPDATE procedure updates (writes) physical Record

```
> PROCEDURE UPDATE
> BEGIN
>   STARTLOG ORDER-HDR-DETAIL
>   PUT ORDER-HDR-DETAIL
>   FOR ORDER-LNE-DETAIL
>       BEGIN
>         PUT ORDER-LNE-DETAIL
>       END
>  STOPLOG
> END
```

[redacted]
QX/PROC/P19

19

UPDATE Procedure

```
. Designer Statements
  > SCREEN qkorder
  > FILE   order-hdr-detail PRIMARY
  > FILE   order-lne-detail DETAIL   OCCURS 10
  > ...
  > BUILD  LIST

  > PROCEDURE UPDATE
  > BEGIN
  >  STARTLOG order-hdr-detail
  >  PUT      order-hdr-detail
  >    FOR    order-lne-detail
  >     BEGIN
  >       PUT order-lne-detail
  >     END
  > STOPLOG
  > END
```

PUT verb

. Updates RECORD

... PUT verb interrogates status of RECORD BUFFER ...

... Takes RECORD from BUFFER and applies UPDATE ...

QUICK
APPL. BUILDER

PROCEDURE Verbs & Predefined Conditions

- **PUT verb**
  - Invokes actual output operation, if RECORD STATUS permits
    - Adding of Record
    - Modifyication of Record
    - Deletion of Record
  - Use primarily in UPDATE procedure
    - BACKOUT procedure may be needed if in other procedures
  - PUT [NOTDELETED] filename [AT expression] [RESET]
    - [NOTDELETED]
      - Update only if RECORD STATUS is not DELETEDRECORD
    - [AT expression] - Record #
    - [RESET]
      - Sets Record to initial values after Update

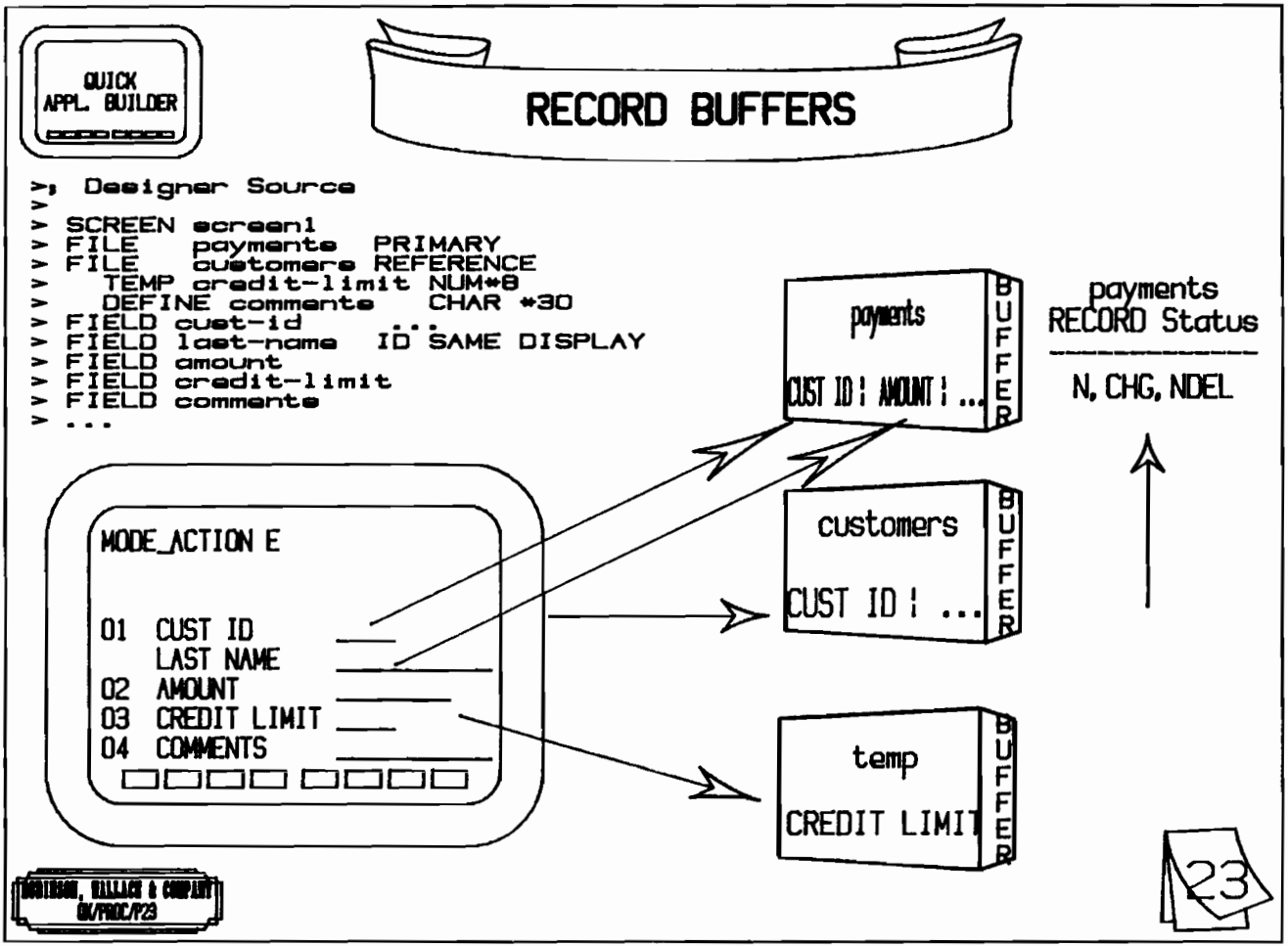
```
> PROCEDURE UPDATE
> BEGIN
>    PUT NOTDELETED ORDER-HDR-DETAIL
>    FOR ORDER-LNE-DETAIL
>          BEGIN
>             PUT ORDER-LNE-DETAIL
>          END
>    PUT ORDER-HDR-DETAIL
> END
```

RECORD BUFFERS

- An area of the Stack that corresponds to the RECORD Layout
  - QUICK stores values entered or found in RECORD BUFFER
  - Values from RECORD BUFFERS are use to UPDATE physical RECORD
- A RECORD BUFFER is created for each FILE
  - Unless OCCURS option is used:
    - RECORD BUFFER for each Occurrence of FILE
- For each RECORD BUFFER QUICK keeps track of status of BUFFER
  1). New RECORD(doesn't exist) or Old RECORD (does exist)
  2). Changed(some value in BUFFER has been changed or added)
      or Unchanged(values are same as in existing RECORD)
  3). RECORD is marked for deletion, or not marked for deletion
- **Predefined Conditions used to test RECORD Status of BUFFER**
  - **NEWRECORD**
  - **ALTEREDRECORD**
  - **DELETEDRECORD**
- Test if RECORD is not new and hasn't been flagged for deletion

```
> IF NOT NEWRECORD AND NOT DELETEDRECORD
>     THEN ...
```

- Initialization of RECORD BUFFER
  - RECORD Status is NEW, UNCHANGED, UNDELETED
- At retrieval of physcial RECORD
  - RECORD Status is OLD, UNCHANGED, UNDELETED


QUICK
APPL. BUILDER
RECORD BUFFERS
>; Designer Source
>
> SCREEN screen1
> FILE payments PRIMARY
> FILE customers REFERENCE
> TEMP credit-limit NUM*8
> DEFINE comments CHAR *30
> FIELD cust-id ...
> FIELD last-name ID SAME DISPLAY
> FIELD amount
> FIELD credit-limit
> FIELD comments
> ...
MODE_ACTION E
01 CUST ID
LAST NAME
02 AMOUNT
03 CREDIT LIMIT
04 COMMENTS
payments
CUST ID | AMOUNT | ...
BUFFER
customers
CUST ID | ...
BUFFER
temp
CREDIT LIMIT
BUFFER
payments
RECORD Status
N, CHG, NDEL
23

Record Status Table

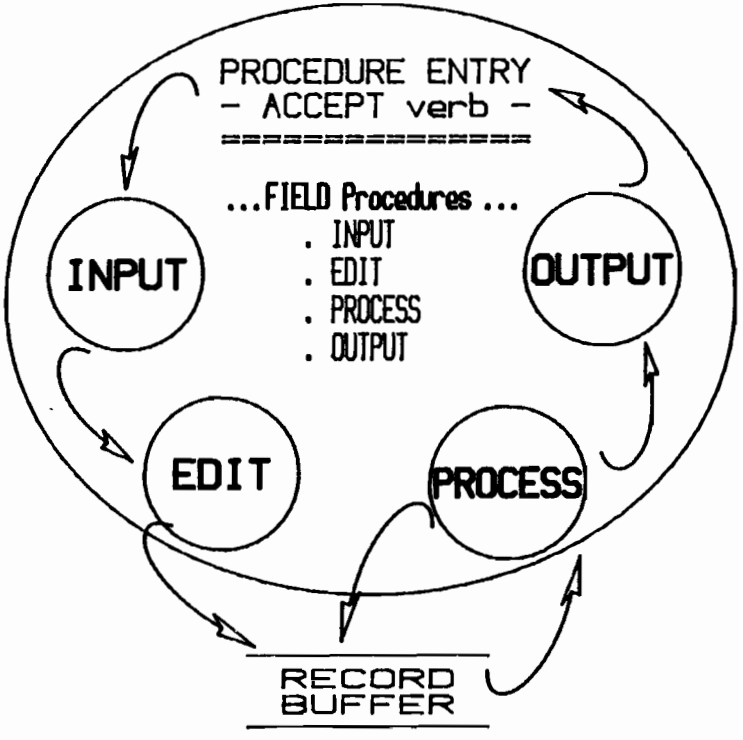
ACTION	TIME	STATUS	Predefined Conditions NEW	ALTERED	DELETED
ENTRY	Initialization	N, UNC, NDEL	T	F	F
	Values Entered	N, C, NDEL	T	T	F
	After Update	O, UNC, NDEL	F	F	F
FIND/ SELECT	Initialization	N, UNC, NDEL	T	F	F
	Retrieval	O, UNC, NDEL	F	F	F
	Values Changed	O, C, NDEL	F	T	F
	After Update	O, UNC, NDEL	F	F	F
DELETE	Flag Deletion	N/O, UNC/C, D	T/F	F/T	T
	After Update	N/O, UNC, D	T/F	F	T

LEGEND

N = NEW
C = CHANGED
O = OLD
UNC = UNCHANGED
D = DELETED
NDEL = NOT DELETED

QUICK
APPL. BUILDER

QUICK Procedure Flow

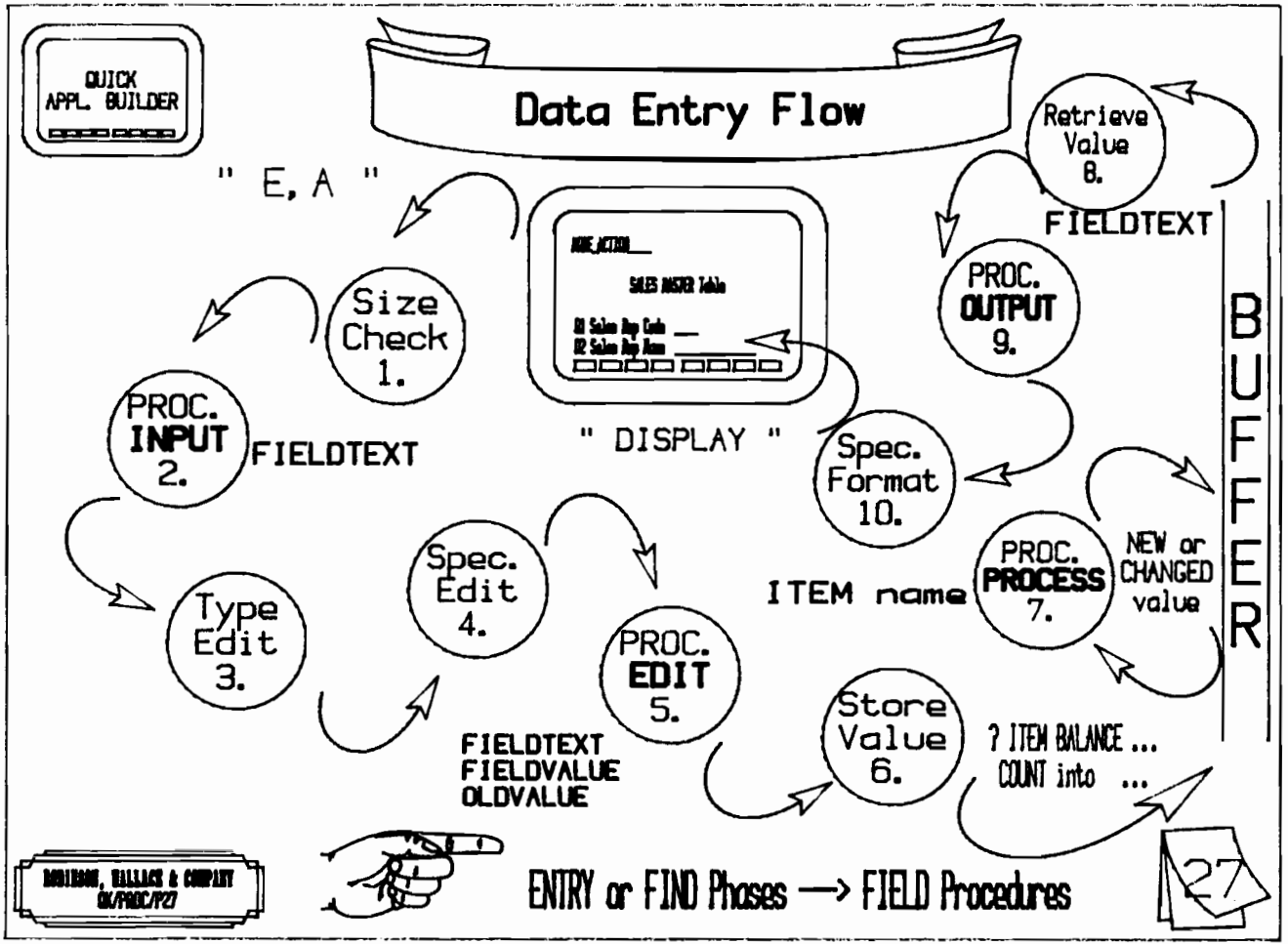


FIELD Procedures

- FIELD Procedures performed by field-related VERBs
  - ACCEPT
  - PROMPT
  - REQUEST
  - DISPLAY
  - EDIT
- FIELD Procedures themselves should not contain FIELD VERBs
- ? Errors encountered in processing of Field VERBs
  - Processing can be halted
  - Messages Displayed
  - User reprompted at FIELD

Message VERBs

Verb	Effect
. INFORMATION	Display text only
. WARNING	UPDATE ---> UPDATE STAY
. ERROR	Processing Stops
. SEVERE	Stops, Reinitialize BUFFERs


QUICK
APPL. BUILDER
Data Entry Flow
" E, A "
Size
Check
1.
PROC.
INPUT
2.
FIELDTEXT
Type
Edit
3.
Spec.
Edit
4.
PROC.
EDIT
5.
FIELDTEXT
FIELDVALUE
OLDVALUE
Store
Value
6.
" DISPLAY "
Retrieve
Value
8.
FIELDTEXT
PROC.
OUTPUT
9.
Spec.
Format
10.
ITEM name
PROC.
PROCESS
7.
NEW or
CHANGED
value
? ITEM BALANCE ...
COUNT into ...
BUFFER
ENTRY or FIND Phases —> FIELD Procedures
27

FIELD Procedures

- Predefined Conditions
  - **FIELDTEXT**
    - Current set of characters entered by user in a FIELD
    - Variable length determined by current contents
  - **FIELDVALUE**
    - Current set of numeric values entered by user from a numeric or date-type FIELD
    - Internal representation same as Dictionary specification
  - **OLDVALUE**
    - Original value of ITEM before any modification or editing

ITEM	PROCEDURE	CONTENTS
FIELDTEXT	INPUT	Operator Input
"	EDIT	New value (character)
"	OUTPUT	Value to be formatted
FIELDVALUE	EDIT	New value(numeric/date)
OLDVALUE	EDIT	Original value

MESSAGE Verbs

- Write to QUICK SCREEN message line
  - Default LINE 24

- **INFORMATION verb**
  - Displays text only
    - > INFORMATION "string" [NOW] [RESPONSE]
    - > INFORMATION n - default to message # QKMSGDES
    - > INFORMATION = expression [NOW] [RESPONSE]
      - [NOW] = Display message immediately
      - [RESPONSE] = Wait for operator response

- **WARNING verb**
  - Changes an UPDATE ---> UPDATE STAY in UPDATE/PREUPDATE
    - > WARNING "string" /n [NOW] [RESPONSE]
    - > WARNING = expression [NOW] [RESPONSE]

- **ERROR verb**
  - Processing stops
    - > ERROR "string" / n
    - > ERROR = expression
  - How QUICK handles an ERROR message depends on PROCEDURE !

- **SEVERE verb**
  - Stop, reintialized Buffer; BACKOUT Procedure
  - In BLOCKMODE will highlight FIELD(s)
  - Position cursor at FIELD associated with error
    - > SEVERE "string" / n
    - > SEVERE = expression

PROCEDURE INPUT field-name

- Modification of Data or Screen Customization before Edits
- INPUT procedure always executed even for "Null Response"
  - Can force execution of EDIT procedure
- One of these predefined conditions is true:
  - ENTRYMODE, CORRECTMODE, FINDMODE, or CHANGEMODE

```
> PROCEDURE INPUT city
> BEGIN
>    IF FIELDTEXT EQ "LA"
>       THEN LET FIELDTEXT = "LOS ANGELES"
>    ELSE IF FIELDTEXT EQ "NY"
>       THEN LET FIELDTEXT = "NEW YORK"
>    ELSE IF FIELDTEXT EQ "SF"
>       THEN LET FIELDTEXT = "SAN FRANCISCO"
> END
```

```
>PROCEDURE INPUT part-no
>BEGIN
>  IF FIELDTEXT EQ "NEWPART"
>     THEN BEGIN
>             RUN SCREEN qkparts MODE E
>          END
> END
```

QUICK
APPL. BUILDER

PROCEDURE EDIT field-name

- Additional editing of FIELD values
- Invoked in response to new or changed values from:
  - ACCEPT or EDIT verbs
- Different degrees of MESSAGE verbs may be issued:
  - INFORMATION, WARNING, or ERROR
  - SEVERE used mostly in Blockmode Applications
- One of these predefined conditions is true:
  - ENTRYMODE, CORRECTMODE, FINDMODE, or CHANGEMODE

```
> PROCEDURE EDIT qty-requested
> BEGIN
>   IF FIELDVALUE > (qty-on-hand + qty-on-order)
>      THEN ERROR="Insufficient Stock: Available Stock is .." &
>              +  ASCII(qty-on-hand + qty-on-order)
> END
```

```
>PROCEDURE EDIT order-no
>BEGIN
>  GET order-hdr-detail SEQUENTIAL OPTIONAL
>  IF ACCESSOK
>     THEN ERROR " ... RECORD already Exists ... "
> END
```

QUICK
APPL. BUILDER

PROCEDURE PROCESS field-name

- For additional FIELD Processing after all Edits applied
- Invoked in response to new or changed values:
  - ACCEPT or EDIT verbs
- Values stored in Record Buffer
  - For Record and/or Temporary ITEMs
- One of these Predefined conditions is true:
  - ENTRYMODE, CORRECTMODE, FINDMODE, or CHANGEMODE

```
> PROCEDURE PROCESS qty-requested
> BEGIN
>   LET commission = qty-requested * price * com-rate
>   IF qty-requested GE 100
>      THEN LET commission = commission * 1.05; 5% Bonus
> END
```

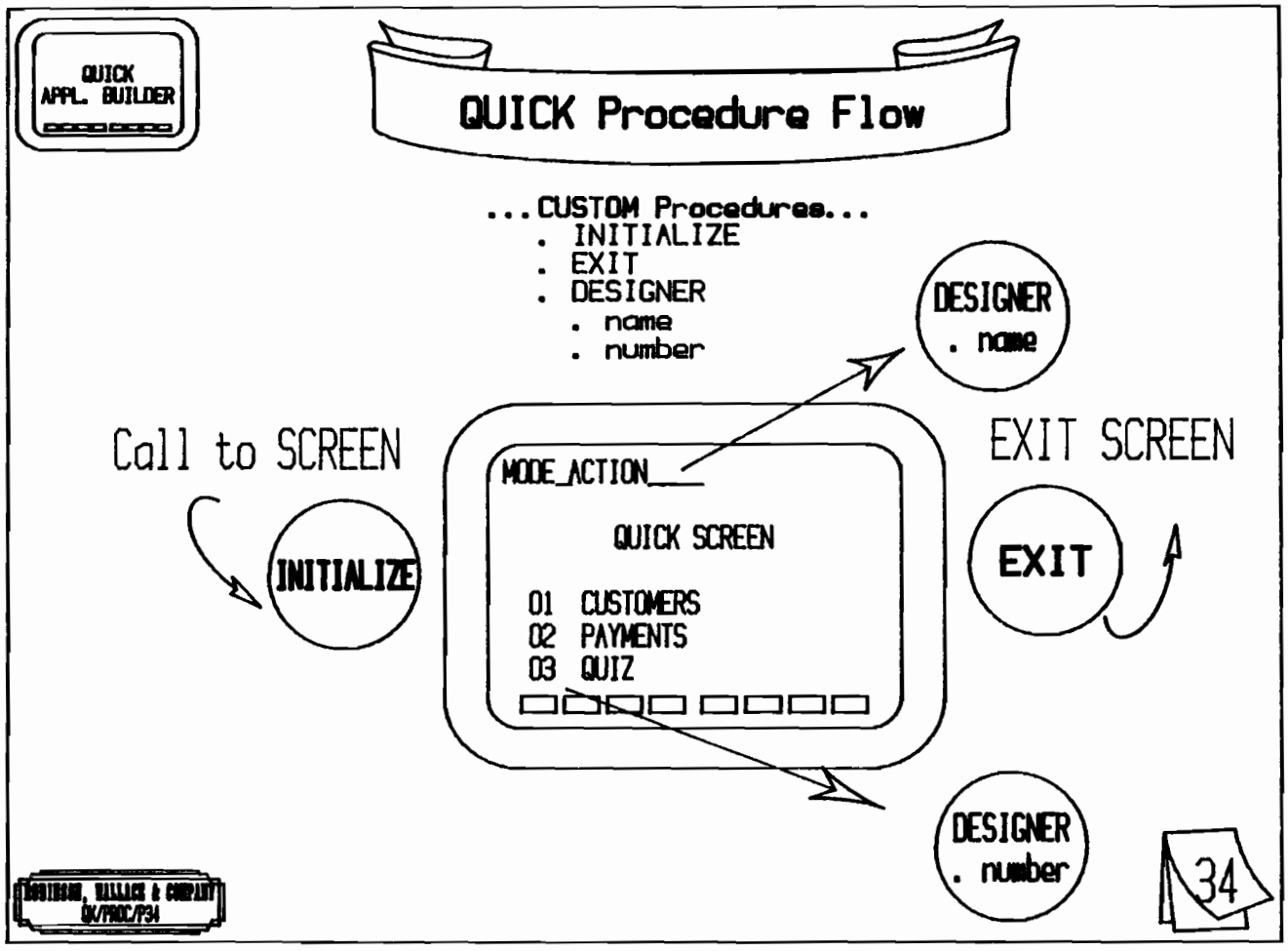
```
>PROCEDURE PROCESS qty-requested
>BEGIN
>  LOCK update-pm
>  GET  update-pm  VIA part-no &
>       USING part-no of line-items
>  LET  qty-on-hand = qty-on-hand of update-pm &
>                   - qty-requested
>  PUT  update-pm
>  UNLOCK
>END
```

PROCEDURE OUTPUT field-name

- After EDITS but before any Formatting options applied
- Invoked in response to new values or retrieval existing values initiated by:
  - ACCEPT, DISPLAY, PROMPT, or REQUEST verbs
- One of these predefined condtions is true:
  - ENTRYMODE, CORRECTMODE, FINDMODE, or CHANGEMODE

```
> ...
> FIELD time-zone SIZE 8    ; X(1)
> ...
> PROCEDURE OUTPUT time-zone
> BEGIN
>   IF FIELDTEXT = "H"
>      THEN LET FIELDTEXT = "HAWAIIAN"
>   ELSE IF FIELDTEXT = "P"
>      THEN LET FIELDTEXT = "PACIFIC"
>...
> END
```

```
>PROCEDURE OUTPUT ddmmyy-date
>BEGIN
>  LET FIELDTEXT = FIELDTEXT[5:2] + &
>                  FIELDTEXT[3:2] + &
>                  FIELDTEXT[1:2]
>END
```


QUICK
APPL. BUILDER
QUICK Procedure Flow
...CUSTOM Procedures...
. INITIALIZE
. EXIT
. DESIGNER
. name
. number
DESIGNER
. name
Call to SCREEN
INITIALIZE
MODE_ACTION___
QUICK SCREEN
01 CUSTOMERS
02 PAYMENTS
03 QUIZ
EXIT SCREEN
EXIT
DESIGNER
. number
34

QUICK
APPL. BUILDER

PROCEDURE INITIALIZE

- Designer Written
- Executed each time SCREEN is called from higher-level SCREEN
  - After ITEM initialization BUT before Screen is Displayed
- ERROR ---> cause PROCEDURE termination
  - EXIT procedure invoked and return to calling Screen
- Can be used for Initialization of Screen Values
  - Security Checks
  - Application Billing

```
> PROCEDURE INITIALIZE
> BEGIN
>   IF NOT MATCHUSER ("MGR")
>      THEN ERROR "... Access Denied to SCREEN ..."
> END

;  Only Users, under Security Class "MGR", may access SCREEN
```

QUICK
APPL. BUILDER

PROCEDURE EXIT

- Designer Written
- Executed when returning to higher-level SCREEN
  - Initiated by:
    - RETURN verb
    - "UR", "^", or "^^" in ACTION
    - "U", "UN", "UR", "US" if AUTORETURN option in use
- ERRORs ---> skip PROCEDURE and return to calling Screen
- One of Predefined conditions is true:
  - ENTRYMODE or FINDMODE

```
> SCREEN ... MENU
> FILE LOGON-APPL   DESIGNER
>      TEMPORARY xtime-logon-beg  NUM *8
> ...
> PROCEDURE INITIALIZE
> BEGIN
>   LET xtime-logon-beg = SYSTIME
> END
```

```
> PROCEDURE EXIT
> BEGIN
>   LET signuser       = SIGNONUSER
>   LET signgroup      = SIGNONGROUP
>   LET signacct       = SIGNONACCOUNT
>   LET date-logon     = SYSDATE
>   LET time-logon-beg = xtime-logon-beg
>   LET time-logon-end = SYSTIME
>   LET term-dev       = PORTID
>   PUT logon-appl
> END

; APPLICATION Billing
```

DODIDSON, WALLACE & COMPANY
QA/PROC/P36

QUICK
APPL. BUILDER

Named DESIGNER Procedure

- Custom Screen Processing
- Save SCREEN space; compared to MENU choice
  - > PROCEDURE DESIGNER name [HELP "string"] [NODATA]
    - name
      - 4 characters maximum
      - Enter in SCREEN ACTION Field
      - "?" in ACTION Field will list all valid entries
    - HELP "string"
      - "??" will display HELP string
    - NODATA ; Data SCREENs only
      - If not specified; Procedure only works if Data on SCREEN

```
> PROCEDURE DESIGNER pwrd NODATA
> BEGIN
>    IF SIGNONUSER = "MGR" or SIGNONUSER = "MANAGER"
>       THEN BEGIN
>               RUN SCREEN pkpword.prod.rwlabs &
>                   CLEAR SCREEN               &
>                   MODE E
>            END
>    ELSE
>       ERROR "... Unauthorized USER ..."
> END
```

QUICK
APPL. BUILDER

Numbered DESIGNER Procedure

- Custom FIELD Processing; instead of default
  - > PROCEDURE DESIGNER nnn
    - nnn
      - Positive integer that matches some FIELD ID
        - PROCEDURE invoked instead of standard action
    - If attached to first ID of CLUSTER repeats automatically

```
> PROCEDURE DESIGNER 01
> BEGIN
>   IF MATCHUSER("Level1")
>      THEN BEGIN
>             RUN SCREEN qkcust.pub.rwlabs   &
>                 CLEAR SCREEN  INPUT C
>           END
>   ELSE IF MATCHUSER ("Level2")
>        THEN BEGIN
>               RUN SCREEN qkcust.pub.rwlabs &
>                   CLEAR SCREEN INPUT B
>             END
>   ELSE
>        ERROR " ... Not authorized USER ..."
> END
```

QUICK
APPL. BUILDER

SCREEN Control VERBs

- **RUN verb**
  - Allow Procedural control of calls to:
    - SUBSCREENs
    - Operating System Commands
    - External Programs
    - UDCs

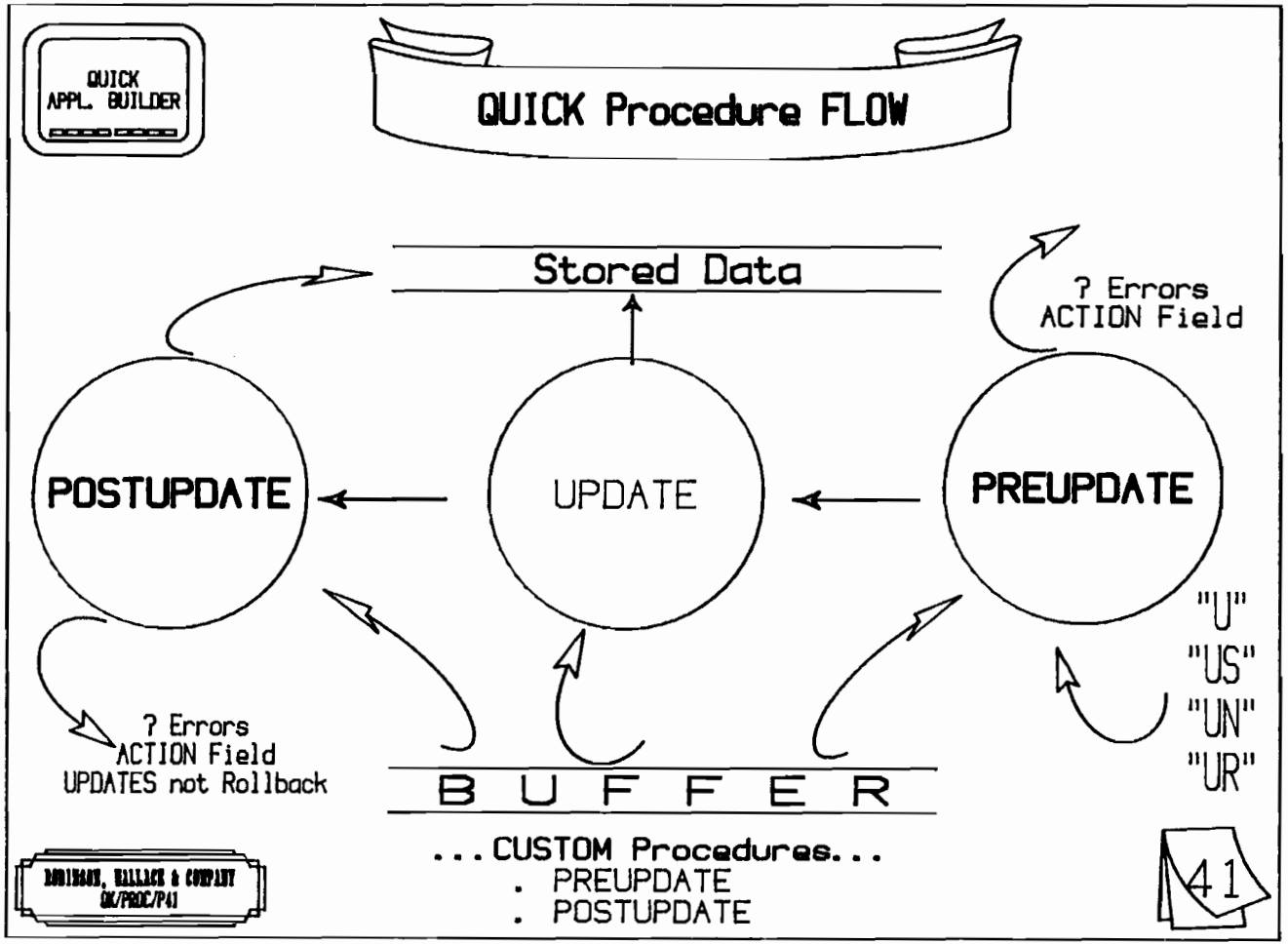
    > RUN SCREEN screen [options]
    > RUN COMMAND "command" [options]
    > RUN COMMAND "run program" [options]

  - Same options as corresponding DESIGN statement, except:
    - AUTO, ID, IF and LABEL

```
> ...
> RUN SCREEN qkpart  PASSING part-number
> RUN COMMAND "PURGE qtrun1"  ON ERROR CONTINUE NOWARN
> RUN COMMAND "QUIZ 'AUTO=qzreport DICTIONARY=dictprod'" &
>     CLEAR SCREEN
> RUN COMMAND "RUN EDITOR.PUB.SYS;SUSPEND" CLEAR SCREEN
> ...
```

SCREEN Control VERBs

- **RETURN verb**
  - Allow SUBSCREEN to return to calling SCREEN under Procedural Control
    - > RETURN
- **CLEAR verb**
  - Blank all or part of Virtural Terminal Memory
  - Use before RUN verb
    - > CLEAR ALL
      - . All Virtual Terminal Memory
    - > CLEAR SCREEN
      - . That part matching what is now displayed
    - > CLEAR LINES n [TO m]
      - . Specific lines only
- **REFRESH verb**
  - Clear and Rewrite portion of Virtual Terminal Memory
  - Use after RUN verb
    - > REFRESH ALL
      - . All of Virtual Terminal Memory
    - > REFRESH SCREEN
      - . That part matching what is now displayed
    - > REFRESH LINES n [TO m]
      - . Specific lines only


QUICK
APPL. BUILDER
QUICK Procedure FLOW
Stored Data
? Errors
ACTION Field
POSTUPDATE
UPDATE
PREUPDATE
"U"
"US"
"UN"
"UR"
? Errors
ACTION Field
UPDATES not Rollback
B U F F E R
...CUSTOM Procedures...
. PREUPDATE
. POSTUPDATE
41

QUICK
APPL. BUILDER

PROCEDURE PREUPDATE

- Designer Written
  - Interfield Editing
- Executed ---> "U", "UN", "US", or "UR" in ACTION
  - For AUTOUPDATE Screens eg... BLOCKMODE
- Message verbs are used to:
  - WARNING UPDATE ---> UPDATE STAY
  - ERROR or SEVERE ---> QUICK disallows UPDATE
- ERRORs ---> PREUPDATE & UPDATE procedures skipped
- One of these predefined conditions is true:
  - CHANGEGMODE or CORRECTMODE

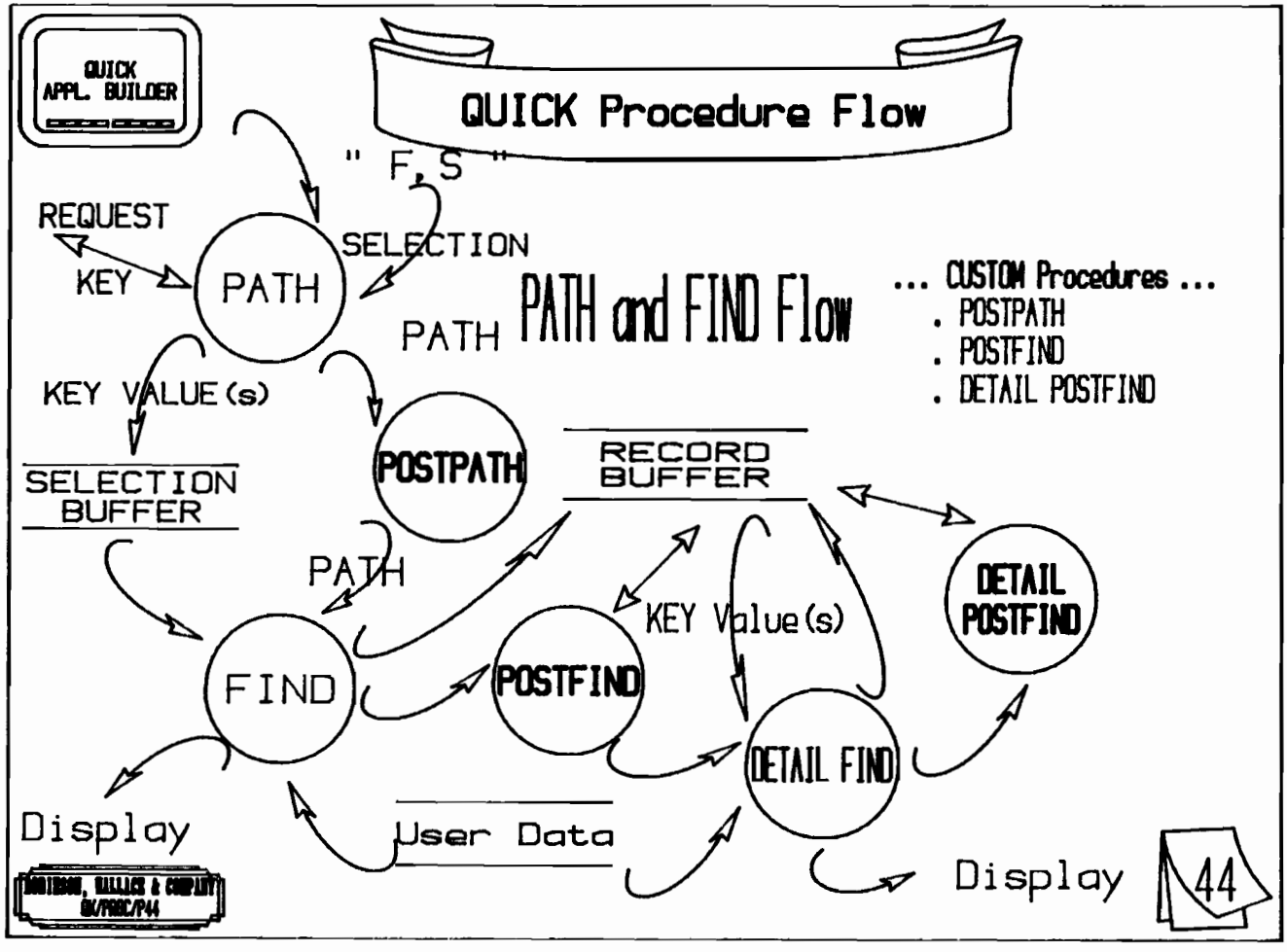
```
> PROCEDURE PREUPDATE
> BEGIN
>   IF trans-count NE batch-count
>     THEN ERROR "... Transactions Out of Balance ..."
> END
```

QUICK
APPL. BUILDER

PROCEDURE POSTUPDATE

- Designer Written
- Executed ---> after UPDATE procedure or AUTORETURN option
- ERRORS ---> PROCEDURE is skipped
  - Any updates applies will not be "Rolled Back"
- One of these predefined conditions are True:
  - CHANGEMODE or CORRECTMODE

```
> PROCEDURE PREUPDATE
> BEGIN
>   LOCK Next-order-num
>   GET  Next-order-num   USING 0
>   LET  Order-no  =  Next-order
>   LET  Next-order = Next-order + 1
>   PUT  Next-order-num
>   UNLOCK
> END

> PROCEDURE POSTUPDATE
> BEGIN
>  INFO = " ... Generated Order-no is ... " + ASCII(Order-no) &
>         NOW  RESPONSE
> END
```


QUICK
APPL. BUILDER
QUICK Procedure Flow
" F,S "
REQUEST
KEY
PATH
SELECTION
PATH
PATH and FIND Flow
... CUSTOM Procedures ...
. POSTPATH
. POSTFIND
. DETAIL POSTFIND
KEY VALUE(s)
SELECTION
BUFFER
POSTPATH
RECORD
BUFFER
PATH
KEY Value(s)
DETAIL
POSTFIND
FIND
POSTFIND
DETAIL FIND
Display
User Data
Display
44

QUICK
APPL. BUILDER

PROCEDURE POSTPATH

- Designer Written
- Executed after PATH Procedure, after any Selection Values are entered in SELECT MODE, and before the Execution of FIND Procedure
- ERRORs ---> PROCEDURE terminates, skips FIND Procedure
- Use instead of modifying PATH procedure
- Predefined condition FINDMODE is True

```
> PROCEDURE POSTPATH
> BEGIN
>   IF PATH = 2          ; Sequential - No Key Values Entered
>      THEN BEGIN
>             IF NOT SIGNONUSER = "MGR"
>                THEN ERROR ".. Sequential Access Not Allowed .."
>           END
> END
```

QUICK
APPL. BUILDER

PROCEDURE POSTFIND

- Designer Written
- Executed after successful completion of FIND Procedure and before display of Retrieved Data
- Used instead of modifying FIND Procedure
- ERRORS ---> PROCEDURE skipped and QUICK prompts at ACTION without displaying Retrieved Data
- Predefined condition CHANGEMODE is TRUE

```
> PROCEDURE POSTFIND
> BEGIN
>   IF current-balance GE 100000;   implied decimal $1000.00
>     THEN INFO " ... Potential Credit Risk: ? Payments ..." &
>          NOW RESPONSE
> END
```

PROCEDURE DETAIL POSTFIND

- Designer Written
- Executed after successful completion of DETAIL FIND Procedure and prior to display of Retrieved Data
- Used instead of modifying DETAIL FIND Procedure
- ERRORS ---> PROCEDURE skipped and QUICK prompts at ACTION after displaying Retrieved Data by the FIND or DETAIL FIND Procedures
- Predefined condition CHANGEMODE is TRUE

```
> ...
> FILE order-hdr-detail  PRI
> FILE order-lne-detail  DETAIL OCCURS 5
>      TEMPORARY rec-cnt NUM *3
> ...
> PROCEDURE DETAIL POSTFIND
> BEGIN
>   FOR order-lne-detail
>       BEGIN
>         LET rec-cnt = rec-cnt + 1
>       END
>   INFO=ASCII(rec-cnt) +"... Line ITEMS for ORDER#" &
>       +ASCII(order-no)
> END
```

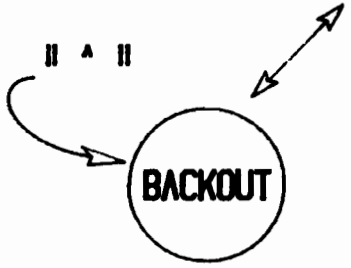
QUICK
APPL. BUILDER

QUICK Procedure Flow

... CUSTOM Procedures ...

- BACKOUT
- INTERNAL

User Data



INTERNAL Procedure —> Subroutine of Procedural Code

QUICK
APPL. BUILDER

PROCEDURE BACKOUT

- Designer Written
- Executed ---> "^" in ACTION
- Usually required to undo PUT verb not specified in UPDATE Procedure
- ERRORS ---> PROCEDURE skipped and the "Backout" Action is continued
- One of these predefined conditions is True:
  - CHANGEMODE or CORRECTMODE

```
> ...
> FILE parts-master DESIGNER ALIAS update-pm OPEN 1
> ...
> PROCEDURE PROCESS qty-requested
> BEGIN
>   LOCK update-pm
>   GET update-pm VIA part-number &
>       USING part-number of order-line-detail
>   LET qty-requested of update-pm = qty-on-hand of ... &
>                                  - qty-requested
>   PUT update-pm
>   UNLOCK
> END
```

```
> PROCEDURE BACKOUT
> BEGIN
>   LOCK update-pm
>   FOR order-line-detail
>       BEGIN
>         GET update-pm VIA part-number &
>             USING part-number of order-line-detail
>         LET qty-on-hand of update-pm = qty-on-hand of ...&
>                                      + qty-requested
>         PUT update-pm
>       END
>   UNLOCK
> END
```

QUICK
APPL. BUILDER

INTERNAL Procedure Example

```
> PROCEDURE INTERNAL proc-incur-loss
> BEGIN
>   LET loss-adj-payroll =  CEILING(paymod * loss-incurred / 100)
>   DISPLAY loss-adj-payroll
> END

> PROCEDURE INTERNAL proc-law-mod
> BEGIN
>   LET adj-losses = loss-adj-payroll * law-bene-mod / 100
>   DO INTERNAL ...
>   DISPLAY adj-losses
> END

> PROCEDURE PROCESS prior-payroll
> BEGIN
>   LET payroll-mod = ceiling(total-payroll/prior-payroll * 100)
>   DISPLAY payroll-mod
>   DO INTERNAL proc-incur-loss
>   DO INTERNAL proc-law-mod
> END

> PROCEDURE PROCESS loss-incurred
> BEGIN
>   DO INTERNAL proc-incur-loss
>   DO INTERNAL proc-law-mod
> END
```

" Routines "

Predefined Conditions

- **Testing PROCESSING MODES**
  - Many Procedures, such as Field Procedures can be invoked from several points in SCREEN processing. Field Procedures include:

    -----> INPUT EDIT PROCESS OUTPUT

    - **ENTRYMODE predefined condition**
      - PROCEDURE ENTRY
      - PROCEDURE APPEND
      - Procedures called by the above
    - **CORRECTMODE predefined condition**
      - After PROCEDURE ENTRY during CORRECTION Phase
    - **FINDMODE predefined condition**
      - PROCEDURE PATH
      - PROCEDURE POSTPATH
      - PROCEDURE FIND
      - PROCEDURE DETAIL FIND
      - Procedures called by the above
    - **CHANGEMODE predefined condition**
      - PROCEDURE POSTFIND
      - PROCEDURE DETAIL POSTFIND
      - After PROCEDURE FIND
      - After CORRECTMODE / UPDATE STAY [US]

QUICK
APPL. BUILDER

QUICK Procedure Flow

. Performance Consideration
  . Code the PROCEDURES in following order:

- . INPUT
- . EDIT
- . PROCESS
- . OUTPUT

- . PATH
- . POSTPATH
- . FIND
- . POSTFIND
- . DETAIL FIND
- . DETAIL POSTFIND

- . PREUPDATE
- . UPDATE
- . POSTUPDATE
- . BACKOUT

- . APPEND
- . ENTRY

- . DELETE
- . DETAIL DELETE

- . INITIALIZE
- . EXIT

- . DESIGNER

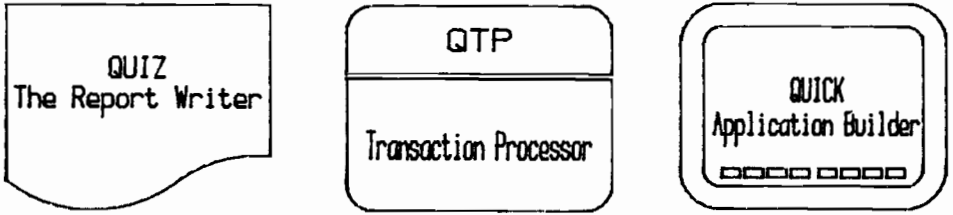
Biographical Summary

David G. Robinson is a principal of Robinson, Wallace & Co., a Los Angeles based firm, providing both consulting and training expertise in the PowerHouse products. David has a strong applications background and is considered a leading authority in PowerHouse. He has over 15 years of data processing experience with the last 5 years working exclusively with the PowerHouse software. First introduced to PowerHouse while serving as a product consultant for Cognos Corporation, then Quasar Systems, was instrumental in assisting sales rep in Los Angeles office to become the first sales person in Cognos history to surpass ONE MILLION dollars in PowerHouse Sales. He also initiated the PowerHouse educational program for Southern California.

Since then he has formed the above named company, developed a dynamic "training" curriculum in PowerHouse, and has provided independant and objective PowerHouse services world-wide.

POWERHOUSE: Performance "Tips & Techniques"

David G. Robinson
ROBINSON, WALLACE & COMPANY
11693 San Vicente Blvd
Suite 168
Los Angeles, CA 90049



ABSTRACT

PowerHouse: Performance "Tips and Techniques"

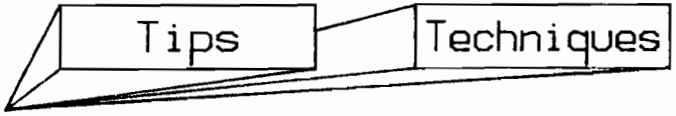
PowerHouse is one of the most widely used 4GL's in the world. The family of PowerHouse products includes a report writer [QUIZ], a Screen Application Builder [QUICK], and a Transaction Processor [QTP], all working under the control of a centralized data dictionary.

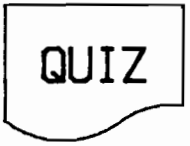
How these programs will affect overall system performance when used in a haphazard manner or without good programmer intervention is the central theme of this paper. This paper will discuss tips on how, when, and where to use these dynamic products to best obtain the maximum through-put with the minimum amount of impact on overall system performance.

The paper will introduce some programming techniques in the various products and discuss their use in the day to day operations of a PowerHouse Environment.

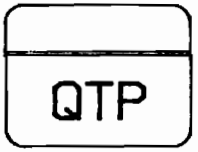
Table of Contents

POWERHOUSE: Performance "Tips & Techniques"



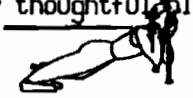


Operating Environment

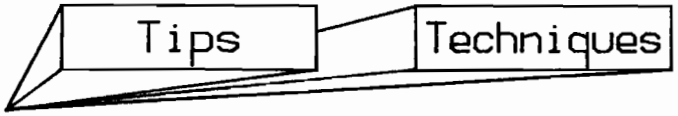


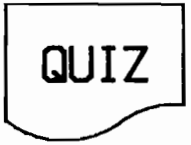
. Data Structures

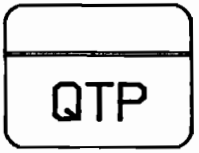
  . DBMS/ HP IMAGE
    . Blocking Factor
      . Optimize to handle SYNONYMS efficiently
    . Capacities
      . Masters
        . 20% Free space for potential growth
        . Beware of Prime Capacities, except for Numeric Keys
      . Details
        . Best to keep low
    . Buffspecs
      . $SET BUFFSPECS can fine tune this
    . Allocation of data sets
      . I/O decrease by thoughtful placements



Data Structures will AFFECT Performance !







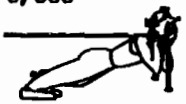
Operating Environment

. Application Usage

. DBMS/ HP IMAGE
. SERIAL vs RANDOM ACCESS
. If large blocking factor may be more efficient to read each data set serially vs reading master serially and doing chained reads on detail datasets

. Processing Master and Detail
. Dependant on number of records, may be more efficient to read detail first serially, and then a random read of master

eg... Master = 50,000 Blocking Factor 10
Detail = 5,000



Poor Data Base Design will AFFECT Performance !

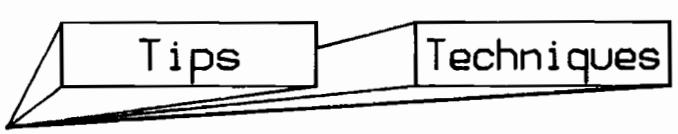
Performance "Tips & Techniques"

QUIZ
The Report Writer

QTP

Transaction Processor

- Linkage Considerations
- Record Selection
- Conditional Expressions
- Sorting Options
- Miscellaneous
- Other Software Alternatives



QUIZ

QTP

Linkage Considerations

* Rule of Thumb

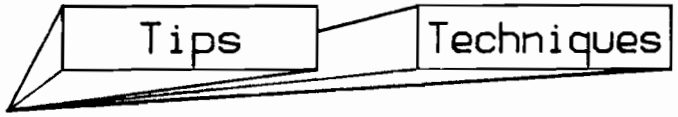
. PRIMARY File should be File with least likely number of hits ---> [Records that may be SELECTed]

. Record Complex (Transaction) is minimized when component can reject records early in ACCESS Linkage Sequence

. DBMS structure may also affect which data set is to PRIMARY File ---> HP/Image or VAX/Rdb
  . For HP/ Image Masters LINK TO their Details ???

. PRIMARY File is the "driver"; determines which records are to be retrieved in ACCESS Linkage Sequence

Incorrect specification of PRIMARY File can cause:
. Additional Reads (I/Os) and Processing Time (CPUs)
. Excessive usage of Disc Space



QUIZ

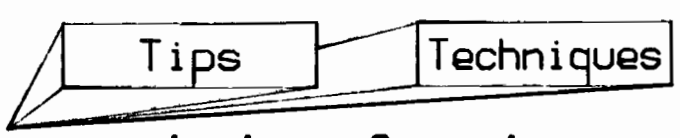
QTP

Linkage Considerations

* Understanding the types of Linkages can provide solutions to complicated QUIZ Reports or QTP Runs:
  - . HIERARCHICAL Linkage
    > ACCESS file1 LINK TO file2 LINK TO file3
  - . PARALLEL Linkage
    > ACCESS file1 LINK TO file2 AND TO file3
    . Implied OPTIONAL

* Correct type of Linkage may also result in elimination of "Duplicate Records"

* The usage of the feature OPTIONAL in the ACCESS Linkage Sequence will affect the number of Record Complexes



QUIZ

QTP

Linkage Considerations

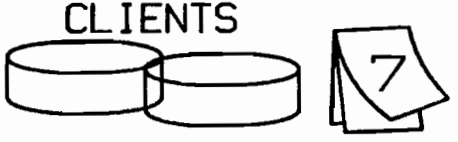
. Significance of PRIMARY File in ACCESS Linkage Sequence
  can result in elimination of sorting large record complexes

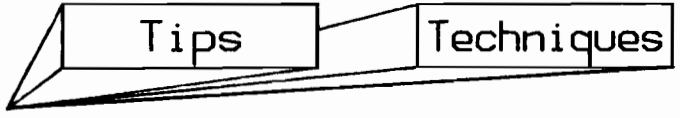
```
  >                    ; HP Example
  > ACCESS clients                                    ; Org INDEXED
  >   LINK    client-id to customer-id of orders      ; Org DETAIL
  >   LINK     order-no to    order-no of line-items; Org DETAIL
* > CHOOSE lastname
**> SORTED ON lastname
  > FOOTING AT lastname ...
```

* Records are retrieved in KEY Ascending Sequence

** Avoid physical sort; but identify control break


```
>; ...  KSAM advantages ...
> CHOOSE keyname ; no value(s)
```





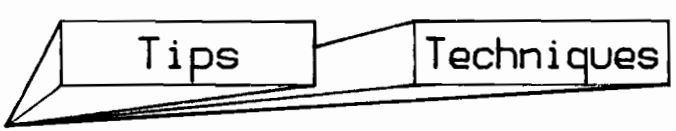
QUIZ

QTP

Linkage Considerations

* HP, VAX, and DG data structures need to be taught to End-Users
  - . Relationships of records
  - . Usage of KEY Retrievals
  - . Effect of Serial Reads

4GLs are Easy to Use & Abuse !



QUIZ

QTP

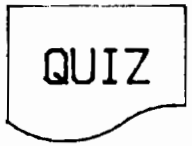
Record Selection

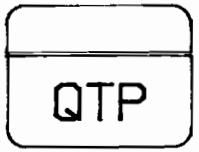
* Use KEY Access whenever possible
  > CHOOSE keyname ...

* For HP/ KSAM Files ; ORG INDEXED
  . GENERIC [Partial Key] Retrievals allowable for CHARACTER Keys
    > CHOOSE lastname "ROB@"
      . Retrieve all ROBs to end of R's
    > CHOOSE lastname "ROB@@"
      . Retrieve all ROBs to end of Z's

* CHOOSE with SELECT IF
  > ACCESS orders
  > SELECT IF date-placed GE 860101
  > CHOOSE order-no 101 102 145 150 160
  > ...

KEY read Selection First ...
Then SELECT is applied ...





Record Selection

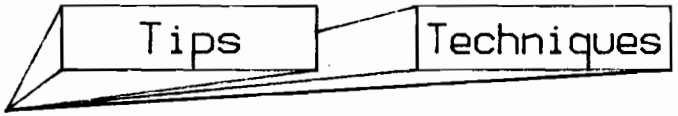
* SELECT IF ; SEQUENTIAL Reads

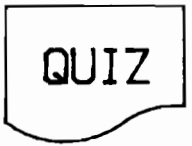
> ACCESS clients
> SELECT IF state EQ "CA" and current-balance GT 0
. AND conditions: Specify "least likely condition" First !

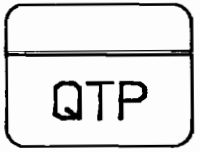
> SELECT IF state EQ "CA" or state EQ "NY" or state EQ "AZ"
. OR conditions: Specify "most likely condition" First !

> SELECT IF 0 NE INDEX("CA*NY*AZ", state)
. More Efficient than previous example

Using INDEX Function is Faster ...







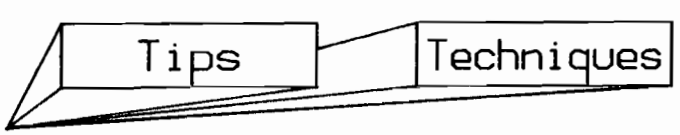
Record Selection

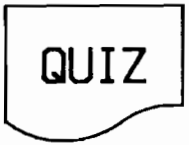
* SELECT IF [NOT] RECORD filename EXISTS
  . Test for Partial Record Complex

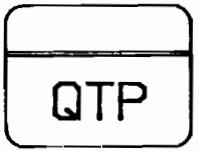
> ACCESS customers LINK to orders and to payments OPTIONAL
>; Parallel Linkage

? Inactive Customers
- . No order activity
- . No payments

> SELECT IF NOT RECORD orders EXISTS &
> AND NOT RECORD payments EXISTS
  . More Efficient than testing individual RECORD ITEMs



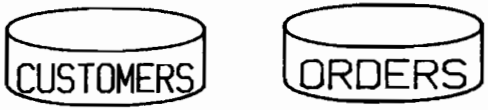




Record Selection

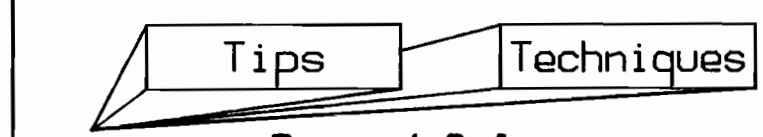
* **SELECT filename IF**
  - . Fewer I/Os
  - . Usage of Stack Space reduced
  - . More Efficient in Multiple File Linkages

> ACCESS customers LINK to orders



> SELECT IF state EQ "CA" AND date-placed GE 860101
> ...
> SELECT customers IF state EQ "CA"
> SELECT IF date-placed GE 860101
  . **Selection is more Efficient**

SELECT filename IF



QUIZ

QTP

Record Selection

ORDERS

C001 860101 ...
C001 860201 ...
C001 860301 ...

CUSTOMERS

C001 SMITH ...

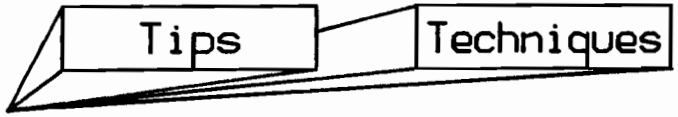
PAYMENTS

C001 750 ...
C001 25 ...

*** SELECT IF vs SELECT filename IF**

> ACCESS customers LINK TO orders AND TO payments
> SELECT IF amount GE 500
Results = C001 SMITH ... C001 860101 ... C001 750 ...
1 Record Complex

> SELECT payments IF amount GE 500
RESULTS = C001 SMITH ... C001 860101 ... C001 750 ...
C001 SMITH ... C001 860201 ... no record
C001 SMITH ... C001 860301 ... no record
3 Record Complexes



QUIZ

QTP

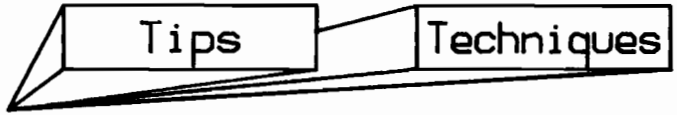
Conditional Expressions

* IF ... ELSE vs Case Processing

```
> ; Inefficient coding
> DEFINE region-cost ZONED *2 = 51 IF city-cde = 30  &
>                          ELSE  52 IF city-cde = 32  &
>                          ELSE  53 IF city-cde = 33  &
>                                   OR city-cde = 34  &
>                                   OR city-cde = 35

> ; Efficient  coding  using Case Processing
> DEFINE region-cost ZONED *2 = CASE of city-cde     &
>        WHEN 30 : 51                                &
>        WHEN 32 : 52                                &
>        WHEN 33, 34, 35 : 53
```

DODIESON, ELLLICE & COMPANY
V 3.0 2/87 PH/PROD/P15

... Case Processing Faster ...



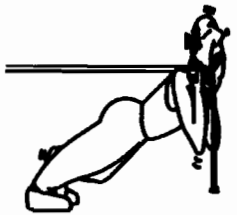
QTP

Conditional Expressions

* **Values of DEFINEd items are calculated for each Transaction**
  . DEFINEs are evaluated prior to record selection using SELECT

* Each DEFINE is evaluated each time it is referenced

```
> DEFINE x CHAR  *2 =  glacctno[8:2]
> DEFINE y ZONED *2 =  01 IF x EQ "CC"  AND amount GE 10000 &
>                  ELSE  02 IF x EQ "BB"  AND amount LE 500   &
>                  ELSE  99
> DEFINE z ZONED *2 =  50 IF x NE "XX"   &
>                        AND y NE  99
> ; x and y are re-evaluated for defined ITEM z
```



... DEFINEs affect PERFORMANCE ...

Tips Techniques

QTP

Conditional Expressions

. For QTP: Input Phase vs Output Phase

```
> ; Inefficient coding
> ACCESS ...                          ; Input  Phase
> DEFINE  x  CHAR  *2 = glacctno[8:2]
> DEFINE  y  ZONED *2 = 01 IF x = "BB"  AND amount GE 10000 ...
> DEFINE  z  ZONED *2 = ...
> SUBFILE ...                         ; Output Phase

> ; Efficient coding
> ACCESS ...                          ; Input  Phase
> TEMPORARY x CHAR * 2                ; Output Phase
>   ITEM x = glacctno[8:2]
> TEMPORARY y ZONED * 2
>   ITEM y = 01 if x = "BB" AND amount GE 10000 ...
> TEMPORARY z ZONED * 2
>    ITEM z = ...
> SUBFILE ...
```

... OUTPUT Phase ...

QTP

Conditional Expressions

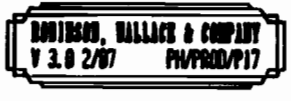
```
**********************************
*        BENCH  MARK             *
*     RECORDS = 13,353           *
**********************************
```

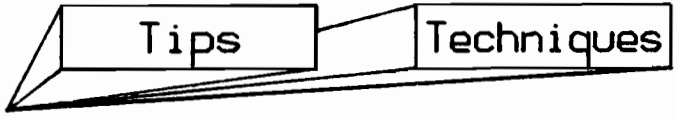
; Inefficient Coding

```
> ACCESS ....                    ; INPUT Phase
> DEFINE x CHAR *2  = glacctno[8:2]
> DEFINE y ZONED *2 = 01 IF x = "BB" AND ...
> DEFINE z ZONED *2 = 50 IF x NE "XX" AND y ne 99
> ; approximately 13 DEFINEs
> SUBFILE ...                    ; OUTPUT Phase

> :EOJ
  CPU SEC. = 2050
```

; Efficient Coding

```
> ACCESS ...                     ; INPUT Phase
> TEMP  x CHAR *2                ; OUTPUT Phase
>   ITEM X = glacctno[8:2]
> TEMP  y ZONED *2
>   ITEM y = 01 IF x = "BB" and ...
> TEMP  z ZONED *2
>   ITEM z = ...
> SUBFILE ...

> :EOJ
  CPU SEC. = 250
```





QUIZ

QTP

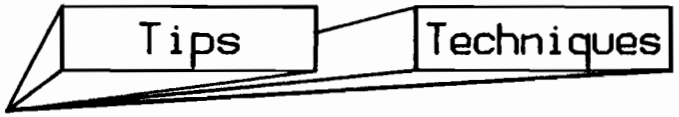
Sorting Options

* Avoiding SORTing Physical RECORDs (Record Complexes)
  . Usage of INDEXED sequential Files as PRIMARY File

```
> ACCESS clients LINK TO ...
> CHOOSE lastname
> SORTED ON lastname
>; Retrieves Records in KEY Ascending Sequence
```

  . Usage of sorted *SUBFILES as PRIMARY File

```
:QTP
> REQUEST backup-sort-sub          ; Daily Backup
> ACCESS customer-master           ; 80,000 records
> SORT ON customer-id
> SUBFILE subcust KEEP INCLUDE customer-id
> ...

:QUIZ
> ACCESS *subcust LINK TO ...
> SORTED ON customer-id
>; PRIMARY File is "driver"
```



QUIZ

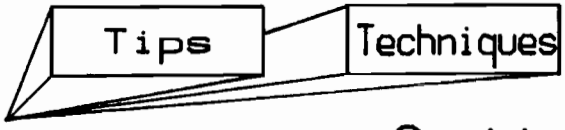
QTP

Sorting Options

* **For Summary type REPORTing avoid unnecessary SORTs**

```
; HP Image DBMS  " Taking advantage of chained reads"
> ACCESS customers LINK TO payments   ; Manual LINK to DETAIL
> SORT  ON customer-id
> FOOTING AT customer-id amt SUBTOTAL ; Any Summary Totals
> FINAL FOOTING COUNT AT customer-id

> ACCESS customers LINK TO  payments  ; Manual LINK to DETAIL
> SORTED  on customer-id
> FOOTING AT customer-id amt SUBTOTAL ; Any Summary Totals
> FINAL FOOTING COUNT AT customer-id

; SORTED on ...  AVOIDs Physical SORTing of RECORDs
```


SORT vs SORTED...



QUIZ

Sorting Options

QTP

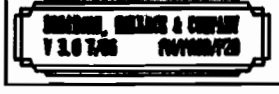
.. Reduce size and NUMBER of Record Complexes before SORTING ...

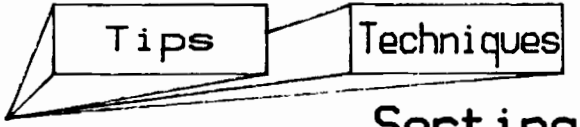
- **Without using SUBFILEs**
  - > ACCESS customers LINK customer-id to customer-id of payments
  - > SORT ON customer-id
  - > REPORT ...

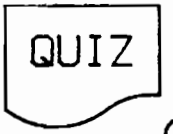
```
C001 SMITH   CA  100  |  C001  CASH 850101  30000
C001 SMITH   CA  100  |  C001  CASH 851016   5000
C001 SMITH   CA  100  |  C001  CASH 851016   2500
          . Sorting 3 Record Complexes
```

- **Using SUBFILEs**
  - **1st pass**
    - > ACCESS customers
    - > SET SUBFILE
    - > SORT ON customer-id
    - > REPORT summary ... [items]
  - **2nd pass**
    - > ACCESS *quizwork LINK customer-id to ...
    - > SORTED ON customer-id
    - > REPORT ...

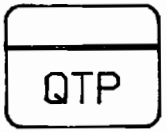
```
C001 SMITH  CA  100
Sorting 1 RECORD
```







Sorting Options



*** Tag Sort Technique**

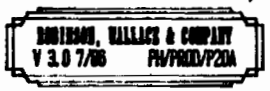
1) Using QTP as front-end to extract multiple SUBFILEs
2) Optionally, create SUBFILE records of all data items
3) Create a SUBFILE of sort key(s) and/or
   * a) matching record numbers in other SUBFILE
     b) enough key info for later retrieval of data from additional files or databases
4) Use TO RECORD (expression) to link sorted keyfile to the unsorted data file

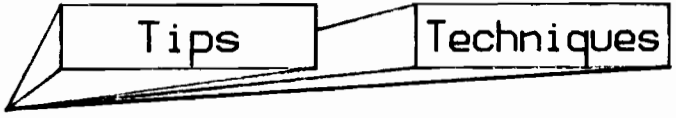
```
:QTP
> REQUEST Extract-Subfiles
> ACCESS customers LINK TO payments
> SUBFILE reptdata INCLUDE ID, lastname, zip, amount ...
>   TEMPORARY rec-no
> SUBFILE sortkeys INCLUDE zip, last-name, rec-no
>   ITEM rec-no  COUNT
>
> REQUEST sorting-sortkeys
> ACCESS  *sortkeys
> SORT ON zip ON lastname
> SUBFILE custkeys INCLUDE sortkeys
```

... LINK TO RECORD ( ) ...

```
:QUIZ
> ACCESS *custkeys LINK &
>  TO RECORD (rec-no) of *reptdata
```

Tag Sort Technique - Example
by Bruce Hobbs
Hobbs, Kramer Enterprises





QUIZ

QTP

Miscellaneous

* **Long-running processes should be batch jobs**

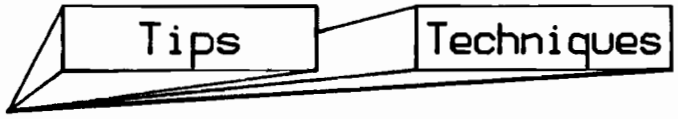
* **Allow QUIZ/QTP during the day only IF:**
  - . CHOOSE is present
  - . Small FILE Size
  - . Urgent priority

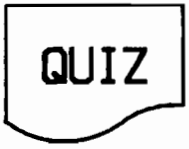
  ---> *"On-line" Screen PERFORMANCE will suffer ...*
  ---> *Terminal is "dead" ...*

* **Compiled vs NOT Compiled**

  PRO: Saves Parser Time

  CON: Need to recompile, when:
  - . Source changes
  - . Dictionary changes
  - . Software Level Changes; new releases



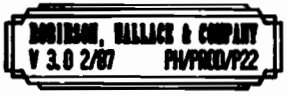


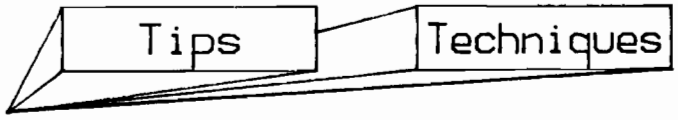
Miscellaneous

* Avoid recompiling "On-Line"
  . Run in Batch and recompile several REPORTs & REQUESTs
  eg... QUIZ REPORTs

```
...
SET DEFAULT
USE     report1.QUIZ
BUILD   report1.PUB
SET DEFAULT
USE     report2.QUIZ
BUILD   report2.PUB
SET DEFAULT
USE     report3.QUIZ
BUILD   report3.PUB
...
```

"On-Line Compiles affect PERFORMANCE"





QUIZ

QTP

Miscellaneous

* **QTP as Pre-pass**

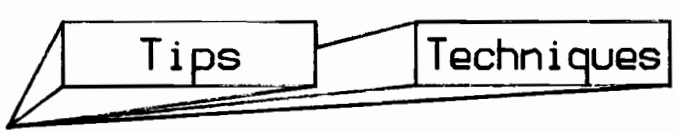
  Using QTP to produce multiple subfiles from one pass of the data base / file(s) to create multiple REPORTs
  - . Different Selection Criteria
  - . Detail Level Reports
  - . Summary Reports

* **For HP/ Image DBMS Users**
  - . Use high-performance version of component
    - . Privileged mode database access is much faster
      QUIZP / QTPP

Alan Perlman
San Diego Housing Commission
SCPUG, 1987

* **For DG/ Users**
  - . Using fast QUIZ & QTP
  - . Minimize use of Repeating KEYs
  - . Usage of INDEX=NO parameter

ROBINSON, WALLACE & COMPANY
V 3.0 2/87 PW/PROD/P23



QUIZ

QTP

Other Software Alternatives

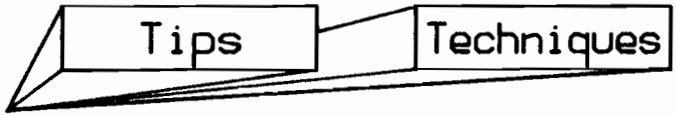
* **SUPERTOOL - Robelle Consulting**
  - Fast Extract from a IMAGE Data SET
  - Allows selection, lookups, sorting

  eg... MANUAL Master 80,000 records
  130,000 capacity

  SUPERTOOL: 6 minutes vs. QTP: 1 hour, 20 minutes

* **QSORT - OPT**
  - optimizes large sorts
  - 1/3 to 1/2 times faster than SORT/3000

* **OMNIDEX - Disc**
  - Record Selection by Multiple Keys w/o Serial Reads
  - Generic (Partial Key) Retrievals
  - Faster than IMAGE DBMS



QUIZ

QTP

Other Software Alternatives

* MPEX - Vesoft
  . Save disc space on QUIZ Report Files.

```
> SET REPORT DEVICE DISC NAME report1
  . QUIZ blocks 1 record per block & record length = 133
```

——> Bad usage of Disc space <——

```
:MPEX     = ALTFILE report1,BLKFACT=BEST
```

——> Increase efficiency of ACCESS; Relocating files <——

```
:MPEX     = ALTFILE oas@.PUB,DEV=NSYSDISC
```

* PDQ for QUIZ - Tymlabs
  . QUIZ Compiler
    . Eliminates interpretive processing during execution
    . Faster than QUIZ - 20% to 50%
  . Future Release Plans
    . Call other routines written in 3GL
    . Other routines can call PDQ/QUIZ procedures

Performance "Tips & Techniques"

QUICK
Application Builder

- QKGO Parms
- Underlying File Structure
- Multiple SCREEN Access
- Miscellaneous

QUICK APPL. BUILDER

Tips Techniques

QKGO Construction & Maintenance SCREENs

HP / : SETQKGO

MODE_ACTION ___

New File _____
Old File _____

01 First screen _____
02 Dictionary file _____

Subscreens:
03 Run time parameter values
04 Action field commands
05 Action and Data field commands
06 Data field commands

Initialize:
302 From existing 3.02/4.00 QKGO file
501 From existing 5.01 QKGO file

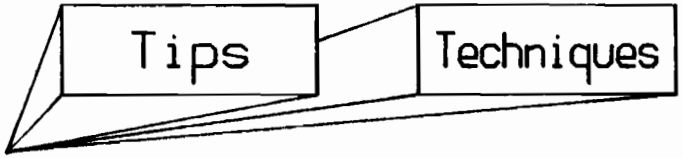
QKGO Parms

Affect the following:
- First SCREEN
- SCREEN Hierarchy & Tables
- EXTERNAL Calls
- Rollback BUFFER
- Operating Environment

Tuning QKGO Parms -->

Controls STACK usage !

Enhance PERFORMANCE !

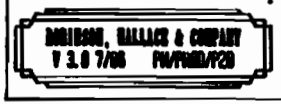


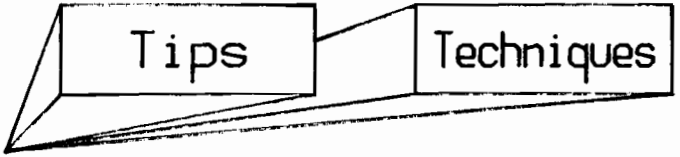
QKGO Parms

* **SCREEN Levels** **[1/5/15] Levels**
  ? Active SCREENs
  ! **Value too high ---> wasted STACK Space**

* **SCREEN Table** **[1/5/50] SCREENs**
  ? maximum no. of SCREENS in APPL. LINES
  or EDS (Extra Data Segments) that are tracked
  ! Value = SCREEN Levels ...
  + more SCREENs can be tracked
  - requires more STACK space

* **Files Opened** **[1/5/100] Files**
  ? maximum no. of Files that QUICK can open
  ! **Value too high --> wasted STACK Space**



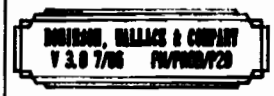


QKGO Parms

* **APPLICATION Lines [5/48/240] Lines**
  ? simulated TERMINAL memory used for stacking SCREENs
  ! **Stacking SCREENs increase QUICK's PERFORMANCE**

* **PROCEDURE Code Pages [2/8/32] 256-byte pages**
  ? pages in memory used for processing Procedures
  + decreases no. of code retrievals
  - requires more STACK space

* **TERMINAL Buffer [80/100/500] Characters**
  ? size of the INPUT/OUTPUT Buffers
  ! Value should never be lower than DEFAULT [100]
  + **increase QUICK's PERFORMANCE**



QUICK APPL. BUILDER

Tips Techniques

QKGO Parms

* ROLLBACK Buffer [128/256/4096] multiples 128 words
  ? size of buffer used to rollback Updates [PUTs]
  + increases QUICK's PERFORMANCE
  - requires more STACK space

* Secondary Blocks [1/32/1000] Blocks
  ? blocks in stack for QUICK to maintain before writting to EDS and/or temporary files in ROLLBACKs
  ! Increase for large Roll Backs

* SEGMENT Size [128/8192/16384] words
  ? size of the EDS (Extra Data Segments)
  ! Value should == size of EDS in System Configuration
  + increase QUICK's PERFORMANCE
  - no effect on Stack Space

QUICK APPL. BUILDER

Tips Techniques

Underlying File Structure

* **Have enough KEYS, but not too many ...**
  - **Too few KEYS**
    - SCREEN will have to do sequential searches or bypass certain retrieved requests entirely
  - **Too many KEYS**
    - Every ADD, DELETE and UPDATE (key change) will be slowed down
    - UPDATEs for non-key items will not be affected
* **Watch blocking factor**
  - **Too Low:**
    - Sequential retrieval (File occurs) will need extra I/O's
  - **Too High:**
    - Direct access will transfer extra (wasted) data from disc into main storage

QUICK
APPL. BUILDER

Tips Techniques

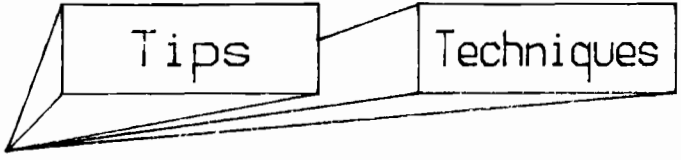
Multiple SCREEN Access

* **Watch multiple SCREEN Access**

  Some techniques that work perfectly from a single terminal can potentially corrupt your data file if multiple operators use the SCREEN at the same time ...

  . **One of these techniques may solve the problem:**

  ---> Lock shared resources
  ---> Give each SCREEN its own private resource, so no sharing is required



Miscellaneous

- **QUICK DETAIL Files**
  - Eliminates calling of subscreen
  - Used in Append Mode Processing

- **Disallowing Sequential ACCESS for inquiries**
  - Designer statement
    ```
    > SCREEN screen1   NOSEQUENTIAL
    ```
  - Procedurally
    ```
    > PROCEDURE POSTPATH
    > ...
    > IF PATH = 2
    >   THEN BEGIN
    >          IF NOT SIGNONUSER = "MGR"
    >          THEN ERROR "... sequential access not allowed ..."
    > ...
    ```

QUICK
APPL. BUILDER

Tips Techniques

Miscellaneous

* **MPEX USER Command**
  . Save programmer time on mass RE-COMPILES
  . Develop naming standards

File: MXUQUICK

```
Start
QUICK
***
File QKSOURCE
SET DEFAULT
SET NOLIST, NOVERIFY
USE  ! QKSOURCE
***
FINISH
EXIT
***
```

%USER MXUQUICK, QK@S
(Assume all QUICK source SCREENs are named QK@S, e.g QKGL001S)
! Use any standard as long as it can be specified by MPEX

POWERHOUSE: Performance "Tips & Techniques"

"Develop a Strategy"

" When you adopt 4GL Technology the pattern of use of your computer changes. If your Company's Procedures & Controls don't change as well poor performance often results ... "

Rick Sodemstrom, Cognos Corp.

4GL's will affect CPU & I/O much more than 3GL's ...
Forte is in saving of Development & Maintenance time ...

Prototyping is feasible with 4GL - " But caution to the wind "

4GL's are Ease to Use & Abuse ...

POWERHOUSE: Performance "Tips & Techniques"

"Develop a Strategy"

* For existing applications determine if PowerHouse is the best "tool" to use ?

  " Realize PowerHouse isn't the perfect tool for all problems "

* For new applications DESIGN with PowerHouse in mind ...

  " Requirements should be clealy defined and understood "
  " Systems should be revieved for efficient coding techniques "
  " Involve 'users' early in development cycle "

* Plan for Software Maintenance

  " Develop standards and program guidelines "
  " Minimize 'tricky' code "

* Product knowledge is a KEY for success

  " End Users need to understand the semantics of their Files "

Biographical Summary

David G. Robinson is a principal of Robinson, Wallace & Co., a Los Angeles based firm, providing both consulting and training expertise in the PowerHouse products. David has a strong applications background and is considered a leading authority in PowerHouse. He has over 15 years of data processing experience with the last 5 years working exclusively with the PowerHouse software. First introduced to PowerHouse while serving as a product consultant for Cognos Corporation, then Quasar Systems, was instrumental in assisting sales rep in Los Angeles office to become the first sales person in Cognos history to surpass ONE MILLION dollars in PowerHouse Sales. He also initiated the PowerHouse educational program for Southern California.

Since then he has formed the above named company, developed a dynamic "training" curriculum in PowerHouse, and has provided independant and objective PowerHouse services world-wide.

ABSTRACT

Improving Back Up Performance

L. Rodoni, Hewlett Packard Company
& **B. McBride,** Hewlett Packard Company

This paper will describe the different bottlenecks encountered by the system during backup. Bottlenecks include the file/disc system, CPU capacity, tape drive speed, and rewind and reload time.

Depending on the bottlenecks various different solutions may be appropriate; various hardware and software options are available.

BLAZE (*) is a software program that has been designed to tackle several of these bottlenecks. It improves access from the fiile system through a technique called 'interleaving' tape drive limitations are addressed through enabling the operation of multiple devices in parallel, and rewind and reload time is eliminated by enabling automatic switching to a second drive after a tape on the first has been wound.

(*) BLAZE is the internal project name. At time of submission of this abstract, the correct name is still being determined through the trademark search process.

USING DBchange TO IMPROVE YOUR DATABASE ADMINISTRATOR'S PRODUCTIVITY

by
Robert Ross
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 UBdelta2 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	Passwords	Passwords
Data items	Data items	Data items
Data sets	Data sets	Data sets
Paths	Paths	Paths
Data set fields	Data set field order	Data set fields
Sort items	Sort items	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	
REVIEW	**COPY**	**PRINT**
Data sets	Data bases	Schemas
Data items		
Paths		
Data set fields		

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
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

← NEW ITEM TYPE

↑ 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 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.

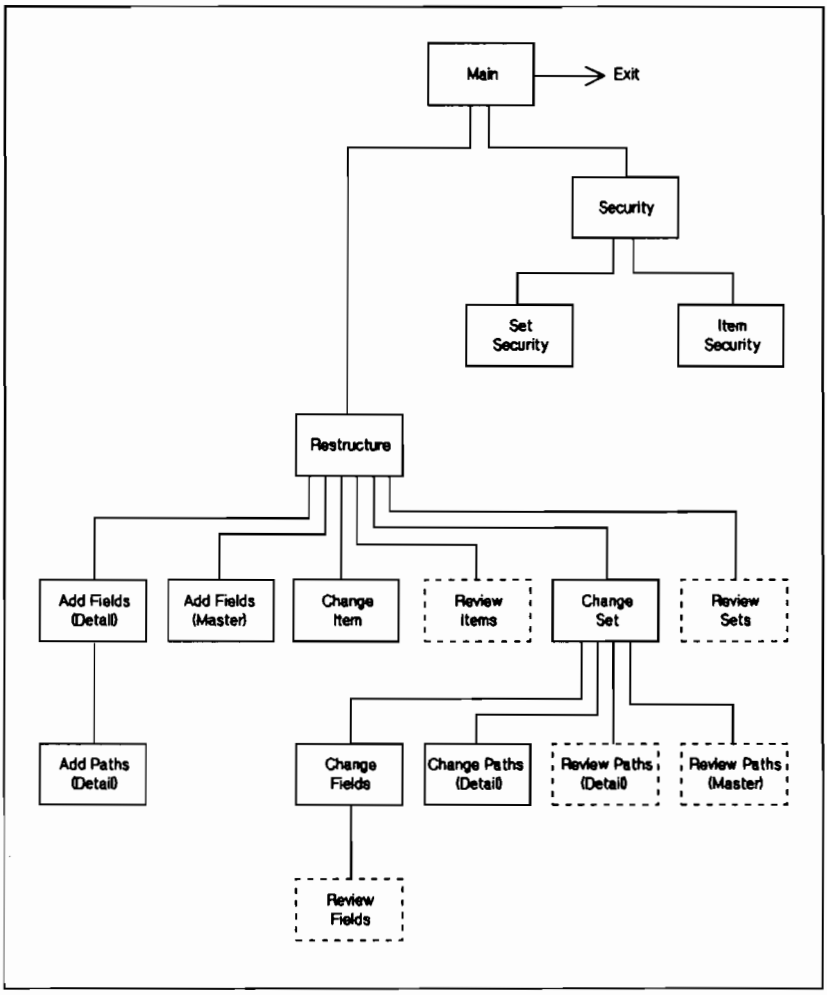


Figure 1 - DBchange screens

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 print the information as if the restructure had already taken place. The user entering the modifications needs to be aware that what he or she is seeing on the review item, set, path, or field screens within DBchange is a combination of what already physically exists in the database plus all changes currently stored in the "change" file.

Checking for Root File Consistency

After entering the name of a database, the first thing DBchange does is check the root file for DBSCHEMA consistency. The root file contains attributes and descriptions of all database objects and relationships, so it is very important that these tables are accurate. These tables also need to be in the same format as DBSCHEMA uses, because when the modifications entered by the user are physically applied to the database, DBSCHEMA is then used to create a new root file that reflects all of these changes. If the current root file is not in the proper format or its tables contain incorrect information, whether from data corruption, modifications by priviledged-mode programs, or hardware errors, then DBchange may pass a schema to DBSCHEMA that will produce a new root file that does not correctly reflect the true state of the database. This situation could produce disastrous results for the user, so a consistency check is performed before DBchange allows the user to enter any changes. The check will take the current root file, create a schema from it, pass this schema to DBSCHEMA to produce a new, temporary root file, then compare the two root files, word for word. If any differences are found, DBchange will warn the user that such a problem exists, then with permission from the user, attempt to repair the problem by storing specific instructions in the "change" file for DBalter, which will accomplish the fix during the restructure. If the problem cannot be fixed, DBchange will again warn the user, saying that it cannot fix the inconsistency, write the differences to a root diagnostic file, then exit. This file's name will be the database name with an "RD" appended to it. Normally, however, DBchange should be able to repair these problems, although this check applies to the root file only. It does not check datasets for corruption.

Change File Capacity

If the root file check passes, the user has the option of making any or all of the changes available. The "change" file has the capacity of storing enough changes to build a database to the maximum number of items, sets, or fields allowed by TurboImage rules. If time doesn't allow for the completion of all changes, the user can exit DBchange and return later to continue adding more entries. The "change" file will accumulate changes, which can be viewed by printing a current schema, until the physical restructuring is done by DBalter.

Schema Efficiency

All of the same concepts used to create an efficient and workable schema apply to DBchange when making changes to a database. For instance, whenever DBchange adds a new item, set, or field, the new addition will automatically default to the last place in the schema or set. Since masters should be placed before any connected details in the schema order, any user adding a new master needs to remember not only to add it to the database, then add fields to it, but also to then move its position in the schema to locate it in front of any details connected to it.

Truncating Data

When alterations are made to existing items, care must be taken to ensure that important information isn't lost. For instance, if the last 10 bytes of an X30 field called "City" don't seem to be used and the administrator decides to trim it to an X20 field, what happens the next time someone wants to do a "FIND" on everyone who lives in "West Death Valley Junction"? (Does anyone really live there?) In this case the user will have to do a find on the first 20 characters, "West Death Valley Ju", which seems somewhat intuitive, but what if you're changing an I2 item to an I1 item and some of the values stored in this field are larger than 64K? DBchange simply truncates the first of the two words, leaving a value that won't be at all intuitive. Fortunately, any time a user tries to do this sort of modification, DBchange will display a "Warning - proceed at your own risk" screen to call attention to the fact that data might be lost if this change is implemented.

Copying vs. Restructuring

Another anomaly of DBchange to take into consideration is that the COPY function is essentially a stand-alone operation. This means that it cannot be combined with any of the other "restructuring" functions. Imagine a situation where a programming team is building a database for use with an application they are developing, and over several days they have accumulated many changes in the "change" file, but haven't physically applied the changes to the database. Now another team wants to make a copy of the current database for their own research and testing, but are unable to do so. Since DBchange considers all changes stored in the "change" file to be part of the existing database structure,

it no longer views the current database as it is, but only as it will be after the physical restructure. To solve DBchange's impasse whether to copy the database as it is physically or as DBchange views it, copying and restructuring were separated into two mutually exclusive functions that share the same "change" file. The result is that all changes in an existing "change" file must be applied to the database before a copy can be made.

Recovering Mistakes

A nice feature of DBchange that can help if you've made a mistake in entering changes is RECOVER. If you have deleted an item, path, set, or field and want to "undelete" it, RECOVER allows you to do just that, unless you've added a new item, path, set, or field with the same name as the deleted one. Or, if you've made a new addition and want to get rid of it without purging all of the changes stored in the change file, simply delete it. However, these "undo's" do not work level by level. For instance, if you delete a current item, then add a new item with the same name, then decide to delete the new item, you are not back at the previous state where you could recover the original item. Once a new addition has been defined with the same name as a deletion, the deleted item, set, path or field is no longer recoverable. As mentioned above, the "change" file contains enough capacity for changes to add the maximum number of items, sets, and fields allowed by TurboImage. However, if one of these objects is deleted, DBchange will keep it in the "change" file in case the user wants to recover it later. This means that a user could enter a series of additions and deletions that would eventually fill the "change" file up, but make very few real additions to the database. In this case, either the "change" file would have to be purged, or DBalter would have to be run to apply the changes to the database.

Running The DBalter Program

The second half of the user interface is a program called "DBalter". While DBchange gathers change information from the user and writes it to the "change" file, DBalter then reads these changes and physically applies them to the database. Although DBchange must be run interactively, DBalter may be run directly from the DBchange menu, or as a session program, or as a batch program. This allows the most flexibility for timing a database restructure, because DBalter requires exclusive access to the database. If DBalter cannot figure out which database to restructure, either through a file equate or by running it directly from DBchange, it will ask you for a name. It will warn you if the database has not been DBSTOREd, and if you give it the go-ahead, it will proceed with reading all of the modifications stored by DBchange in the "change" file and applying them all to the database simultaneously, set by set.

Temporary File Space

Every set that will need restructuring is flagged by DBchange, and DBalter attempts to create temporary files for each of them with enough room to accommodate any additional fields or a change in capacity. If there is not enough room on disc to hold all of these temporary files, DBalter will ask if you want to use the more risky method of creating one temporary file at a time, making changes only to that set, then purging the old set to make room for the next temporary file, or stop at this point and use DBchange to delete some of the changes so fewer sets will be modified and less room will be needed. If the first choice is opted for and a failure occurs during processing, the database will be left in an inconsistent state with some of the sets missing, and will need to be restored from a backup.

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 than 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 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 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.

Table of Contents

"J O B S S Y S T E M"

SUBMITTING JOBS WITH VARIABLE PARAMETERS PROVIDING SECURITY AND CONTROL

JORGE ROUNTREE

PETROLEOS MEXICANOS
3600 SOUTH GESSNER, SUITE 236
HOUSTON, TEXAS 77063

Table of Contents

Chapter 1

INTRODUCTION

Because of the necessity of counting on a better way to operate our 750 programs actually in production, we decided to invest some time to design a system to solve problems like :

1.- Having a lot of separate files where the jobs were defined without any link or sequence among them.

2.- Having a group of jobs that had to be repeatedly modified to include the variable data on each run process.

3.- Having a very experienced operator in MPE commands and the systems of our company to react without mistakes every time a process asked for information.

4.- Having no possibilty to schedule processes that require a parameter to be executed at an exact date and time and without human intervention to define the values.

5.- Having documentation of the jobs not integrated with command files. Therefore, it could become obsolete in days because of the dynamics of the operation.

As a result of this effort, we created what we call **THE JOBS SYSTEM.** Our **JOBS SYSTEM** was designed to allow the operations manager to define jobs with optional variable parameters via menus and screens and to interact with documentation programs to print cross-references of jobs to programs and jobs to data sets.

These jobs are stored in a data base which contains general information such as the system it belongs to, frequency category (subsystem) it runs in, the job name or identifier, the user ID from which it must run, a label description for the operator, a general description for documentation, prompts for the operator to enter values for variable parameters and a job "skeleton" of the commands and parameters necessary to execute the programs.

Every parameter has a pattern, a prompt, a type and a

default value to assist the operator when he submits the job for execution. These parameters can be defined as global parameters used in several locations in the commands file. The system is also capable of sending information such as telephone numbers and password codes to remote job entry programs to create communications and transfer data through telephone lines.

The operations manager can define a validation criteria with the patterns for the variable parameters to assure the correct input of the data. The operator has the option to establish a date and time the job should run which are validated previously to being accepted . As we mentioned, the global parameters allow the operator to be prompted only once for dates or date ranges that are used in several programs included in the job, avoiding mistakes and lazy times in the computer.

Chapter 2

OBJECTIVES

The objective of our **JOBS SYSTEM** was to facilitate the operator's activities, but at the same time make his job more efficient, secure and controlled. In other words, our system has given us the possibility to count on operators who do not require large amounts of training and experience to successfully accomplish their functions. The system allows the operations manager to define the jobs and the operator to execute them with the easy use of menus and screens. Our system was intended to provide security to the jobs by limiting access with the help of a security system also developed by us. We are also able to control the operations by assigning the jobs to frequency categories. The interaction with documentation programs gives us the possibility to print reports used to learn and analyze our current programs in production. We tried to create a system with enough flexability for changes or expansion by utilizing a data base, allowing easy access to the definition of jobs, having the capability of variable and global parameters and permitting variable times of execution. In essence, what we did was to transfer the responsibilty of the repetitive activities from the operator to the computer (JOBS SYSTEM) so that once it learns how to do its job, it does not fail.

Chapter 3

LIMITATIONS

3.1 AT EXECUTION TIME

The **JOBS SYSTEM** has its limitations. For example, the maximum number of variable parameters which the operator can be prompted for in a single job is 14, but we have the facility of using global parameters and all the fixed parameters we want to include. Each label description prompt and variable parameter value is limited to 66 characters. However, the maximum size of the parameter is also dependent on the size required by the individual programs that utilize it.

3.2 AT DEFINITION TIME

Each job is assigned to a system and subsystem which identifies the department it is being executed for and the frequency with which it runs. Although the job system was designed to have an unlimited number of systems, subsystems, and jobs defined in the data base, the screens are currently capable of having a maximum of 32 systems, 32 subsystems per system and 32 jobs per subsystem. Each system and subsystem name is limited to 16 characters and each job name is limited to 8 characters.

The operations manager creates each job into the system by defining the header and the lines to build the "skeleton" with the commands and parameters. The maximum number of jobs that can be currently defined in the system is 9999. Each label used for identification on the operator's job screen has a maximum of 34 characters. A job can be used in more than one system or subsystem but with a different job number. A general description is also defined for documentation purposes with a limit of 4 lines of 60 characters each. The lines of the jobs are also sized to 60 characters per line, but they can be extended to the next line with the continuation signal "\". An unlimited number

of command lines can be defined for a single job number.

Chapter 4

FUNCTIONS AND OPTIONS

The **JOBS SYSTEM** has basically two functions : the definition of jobs and the execution of them, and each one has its own options.

4.1 OF THE OPERATION'S MANAGER

The operations manager is responsible for defining the jobs on the system by using an on-line program handled through SCREEN NO. 1 or 2 that updates the data base. He must define the following characteristics to create a new job :

- System the job is being executed for
- Subsystem or frequency of execution
- User ID of who must execute the job
- Priority of execution
- Job name to identify the job on the screen, standard list and the documentation reports
- Label description for the operator to identify jobs on the jobs menu and to verify them on the parameters screen
- General description to be utilized by the documentation program to generate the reports
- "Skeleton" of the job which includes the command lines and parameters

The system is responsible for automatically assigning a job number taken from the data base, and keeping track of the last one assigned.

The operations manager has the following options when he defines the job :

- Add (A), Delete (B), Display (D), Modify (M), Inactivate (I) or Reactivate (R) a job from the system
- Assign a new system to a job and automatically update the systems table in the data set SISTEMA-MST of the data base DBTBL that contains the list of all valid system names
- Create a new system or subsystem on the menus by

defining a new system or subsystem to a job
- Modify (M) or Delete (B) command and parameter lines individually or Insert (I) an unlimited number of them after any line
- Include as lines of the job comments to assist in analyzing or understanding it
- Include messages to the operator of the status of the job or as instructions to assist him in the execution
- Define parameters as fixed, global or variable
- Define the characteristics, pattern, prompt and default value of the variable parameters
- Define parameters as instructions for a remote job entry program to establish communications between computers with telephone lines and to provide logon procedures to be able to transfer files

4.2 OF THE OPERATOR

When the operator submits a job for execution, he is responsible for doing the following :

- Selecting the system or departament the job is being executed for
- Selecting the subsystem scheduled for the job
- Identifying and entering the job number to be executed from the jobs menu
- Entering the date and time of execution of the job if he does not want immediate execution
- Entering the values of the parameters he is prompted for or simply pressing the key < ENTER > to keep the defaults
- Sending the job for execution with a function key entitled < EJECUTAR JOB >

The system is responsible for the following :

- Finding the password which resides in a protected file for the user ID which must run the job
- Creating the job control card with job name, logon ID, password and priority
- Transfering the job "skeleton" to the file that will be submitted for execution
- Adding the 'end-of-job' command to terminate the execution
- Sending the job to the queue for execution with all its predefined characteristics
- Creating a communication line, if aplicable, to the job

The operator has the following options when he submits a job :

- Execute a job immediately or schedule it for a certain day and hour.
- Use the default values for the parameters
- Print the screen to create some documentation about what he did
- Navigate through the systems, subsystems and jobs menus to select his process to be done
- Erase the screen or cancel the job to begin entering the parameter values again
- Reinitialize the screen with the default values if he has modified them

Chapter 5

SECURITY AND CONTROL

5.1 AUTHORIZED USERS

The **JOBS SYSTEM** interacts with a security system that controls the access to different predefined systems. The screens and menus of the **JOBS SYSTEM** can only be accessed if the logon user ID has been defined with the adequate capacity to be able to use it. Each user ID has defined in the data set CR-USER-MAN of the data base DBTBL an array of capacities that allow him to do various functions in the systems. To update the jobs (which includes displaying, adding, deleting, modifying, inactivating and reactivating) and to execute them, the user ID must have assigned the capacity number 15 defined as "The ability to give maintenance to the jobs and and execute them in operations".

5.2 DEFINED JUST ONCE

The system avoids the operator from having to create or modify the jobs repeatedly or utilize User Defined Commands at execution time. Once the job has been defined in the system and tested, there is no possibility of failing in the definition of the commands because the same job instructions are always being sent to execution. This results in security and efficiency.

5.3 VALIDATED DECISIONS OF OPERATOR

To provide security and control so that the operator does not make mistakes in entering the parameter values, a powerful validation criteria is integrated into the definition of the parameters through the use of PATTERNS. All the values the operator is prompted for are compared to

a predefined PATTERN for validation of type, size, structure and characteristics before being accepted.

5.4 EXECUTED AT EXACT TIME

To schedule the day and time of execution, the jobs are classified into subsystems that define the frequency and time they are run. This secures that the operator does a more efficient job and that the production system is controlled through the computer and not through the dependency of the operator.

Chapter 6

STRUCTURE

6.1 OF THE SYSTEM

The system is divided into two processes as we can see in FIGURE 3, one allows the user to define the characteristics of the job and the other permits him to submit it for execution with variable parameters.

The process utilized by the operations manager to define the job consists of two sections : the header section and the commands and parameters section. The first section of the screen, as described in SCREEN NO. 1, contains the definition of the job header which includes the job number, job name, priority, etc. and the job description. All fields are required and must be entered before a job number is automatically assigned by the system when adding a new job. The second section of the screen, as described on SCREEN NO. 2, allows an infinite number of jobs commands and parameters (with the mentioned limitations) to be entered.

The function to execute the jobs consists of three menus and one screen to enter information. The structure of these processes, as described below, assists the operator in submitting the job to be executed :

1. SCREEN NO. 3 : Prompts for the number of the system
2. SCREEN NO. 4 : Prompts for the number of the subsystem or frequency category
3. SCREEN NO. 5 : Prompts for the number of the job identified by a label description
4. SCREEN NO. 6 : Prompts for the values of the variable parameters of the selected job with the option to specify a date and time of execution

The job is then sent for execution. The program creates the job file including the control card, job skeleton and end-of-job command and streams it to be run in batch immediately or at the specified day and time with its predefined priority. The system reports to the operator the operating system's job identification number, returns to the

jobs screen and informs him of the job's submitting results.

6.2 OF THE INFORMATION

A data base DBTBL as described in FIGURE 1 contains the data of the **JOBS SYSTEM** consisting of five data sets, three details and two masters.

1.- NUM-JOB-MST
List of all job numbers already assigned to assure a unique key
TYPE : Automatic Master
KEY : NUM-JOB
CONTENTS : Job numbers

2.- SISTEMA-MST
List of all system names currently assigned to the jobs
TYPE : Automatic Master
KEY : SISTEMA
CONTENTS : System names

3.- LINEAS-JOBS-DTL
Job skeleton of the command lines and parameters of each job number
TYPE : Detail
KEY : NUM-JOB
LINKS : NUM-JOB-MST via NUM-JOB
CONTENTS : Commands and Parameters
Job number
Line number

4.- DESC-JOBS-DTL
Definition of each job number
TYPE : Detail
KEY : SISTEMA
LINKS : SISTEMA-MST via SISTEMA
NUM-JOB-MST via NUM-JOB
CONTENTS : Job number
Job name
User ID for execution
General description for documentation
Label description for operator
Priority of execution
Active or Inactive state
System to which it belongs
Subsystem or frequency category

5.- TBL-GLOBAL-DTL
Last job number assigned to allow the system to automatically assign the next sequential job number
TYPE : Detail
KEY : None
LINKS : None

CONTENTS : Last job number

In order to create some documentation about the jobs, the system uses another data base (DBDOC) described in FIGURE 2 which contains information about the the programs and data sets cross-referenced with the jobs.

1.- NOMBRE-MST
List of all program names to assure a unique key
TYPE : Automatic Master
KEY : NOMBRE
CONTENTS : Program names

2.- NOM-DSET-MST
Cross-reference between all program names and their associated data sets
TYPE : Automatic Master
KEY : NOM-DSET
CONTENTS : Program name
Data set name

3.- DATA-SET-MAN
Cross-reference betwwen all data set names and their associated data base
TYPE : Manual Master
KEY : DATA-SET
CONTENTS : Data set name
Data base name

4.- PROGS-DTL
Definition of all programs
TYPE : Detail
KEY : NOMBRE
LINKS : NOMBRE-MST via NOMBRE
CONTENTS : Program name
Author
Creation date
General description for documentation
If obsolete
System to which it belongs
Date of cross-reference creation
Language

5.- CROSSREF-DTL
List of all program names with additional information
TYPE : Detail
KEY : NOMBRE
LINKS : NOMBRE-MST via NOMBRE
NOM-DSET-MST via NOMBRE
DATA-SET-MAN via DATA-SET
CONTENTS : Program name
Data set
Program name-Data set name
Read or Write access

Chapter 7

UTILIZATION

7.1 CREATING/UPDATING A JOB

The operations manager utilizes the screen entitled "DESCRIPCION DE JOBS" demonstrated on SCREEN NO. 1 to define the characteristics and description of the jobs. All the screens of the **JOBS SYSTEM** work in block mode and the key labeled < ENTER > must be pressed at the end of all functions. While in the display (D) or modification (M) mode the following function keys are utilized to navigate sequentially through the job records :

```
< SIGUIENTE  REGISTRO > = NEXT  RECORD
< PRIMER     REGISTRO > = FIRST RECORD
< OTRO JOB >            = OTHER JOB.
```

To delete, inactivate and reactivate a job, it must first be displayed on the screen utilizing the "D" function or the function keys mentioned above. Then the desired function ("B" for delete, "I" for inactivate or "R" for reactivate) must be entered and completed with the < ENTER > key. The system will respond with a message if the transaction was successful. The following function key can be utilized to document the job before deleting or inactivating it :

```
< IMPRIMIR PANTALLA > = PRINT SCREEN
```

To add a new record, the function "A" and all the fields must be entered for the system to automatically assign a job number. The < TAB > key is utilized to pass among the fields of the header of the job. If a new system or subsystem is used, it will automatically be added to the systems and subsystems menus. The function key mentioned above to print the screen can also be utilized for documentation for all new jobs. The following function keys can also assist the user in entering the data :

```
< REINICIA PANTALLA >  =  REINITIALIZE THE SCREEN
< BORRAR   PANTALLA >  =  ERASE THE SCREEN
```

The commands and parameters of the job can be defined and updated via the function key < LINEAS DEL JOB > = JOB LINES.

The screen as described in SCREEN NO. 2 displays ten lines of the "skeleton" at a time. The following function key is utilized to display an additional ten lines of the job :

< SIGUIENT LINEAS > = NEXT LINES

While creating lines of a new job, the field labeled "MAS" is marked to receive an additional ten lines to continue. To add new lines to the end of an existing job, the following function key is used :

< AGREGAR LINEAS > = ADD LINES

To update the lines, the letters "M" for modify, "B" for delete or "I" for insert can be placed in the field located before each line to make individual changes. The following function keys are available to return through or exit from the system :

< OTRO JOB > = NEXT JOB
< PANTALLA PREVIA > = PREVIOUS SCREEN
< MENU PREVIO > = PREVIOUS MENU
< TERMINA PROCESO > = END PROCESS

7.2 EXECUTING A JOB

To choose a job to submit for execution the operator navigates through the following menus and selects the corresponding number.

1. SCREEN NO. 3 : Lists all the systems available
2. SCREEN NO. 4 : Lists all the subsystems of a system
3. SCREEN NO. 5 : Lists all the jobs available for a subsystem identifiable by a label

The operator is then sent to another screen, refer to SCREEN NO. 6, to enter the variable parameters of the selected job. He also has the option to specify the date and time of execution in the indicated fields labeled < FECHA > and < HORA > or leave them blank for immediate execution. Once the operator has reviewed his entries, he presses the function key labeled < EJECUTAR JOB > = EXECUTE JOB to submit the job. He then records the operating system's job identification number and verifies if the submission was successful. After the job has finished, the operator receives a standard list labeled with the job name from the printer and verifies that the job was executed without any errors.
The **JOBS SYSTEM** interacts with a documentation program to generate three reports to assist the user in controlling and

analyzing the information.

Chapter 8

REPORTS

8.1 SRP060Z

List of all jobs in the system

SORTED BY : System
Subsystem
Job number
CONTENTS : Job name
Label description
General description
Each line in the job which includes :
1. Commands
2. Parameters
a) Prompt
b) Pattern
c) Defaults
3. Messages to operator

8.2 SRP160Z

List of all programs cross-referenced with jobs

SORTED BY : Program name
CONTENTS : Job number
Job name
Label description
System
Subsystem

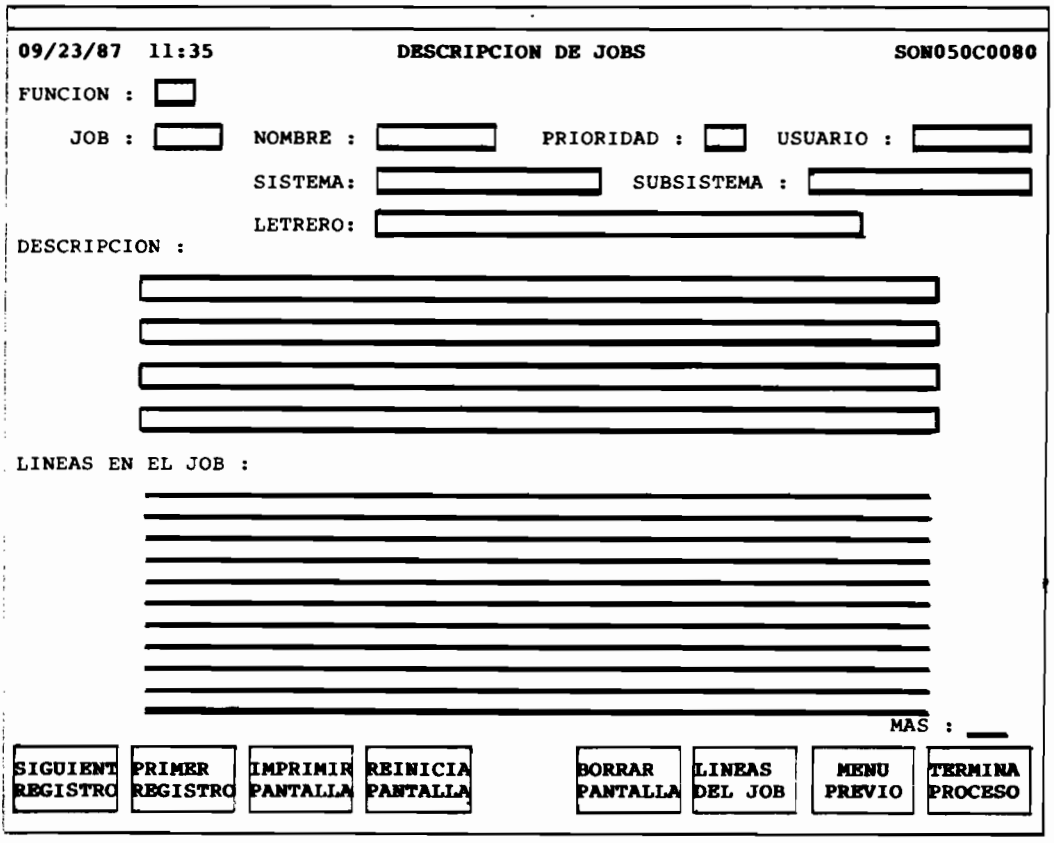
8.3 SRP123Z

List of all jobs cross-referenced with programs and data sets

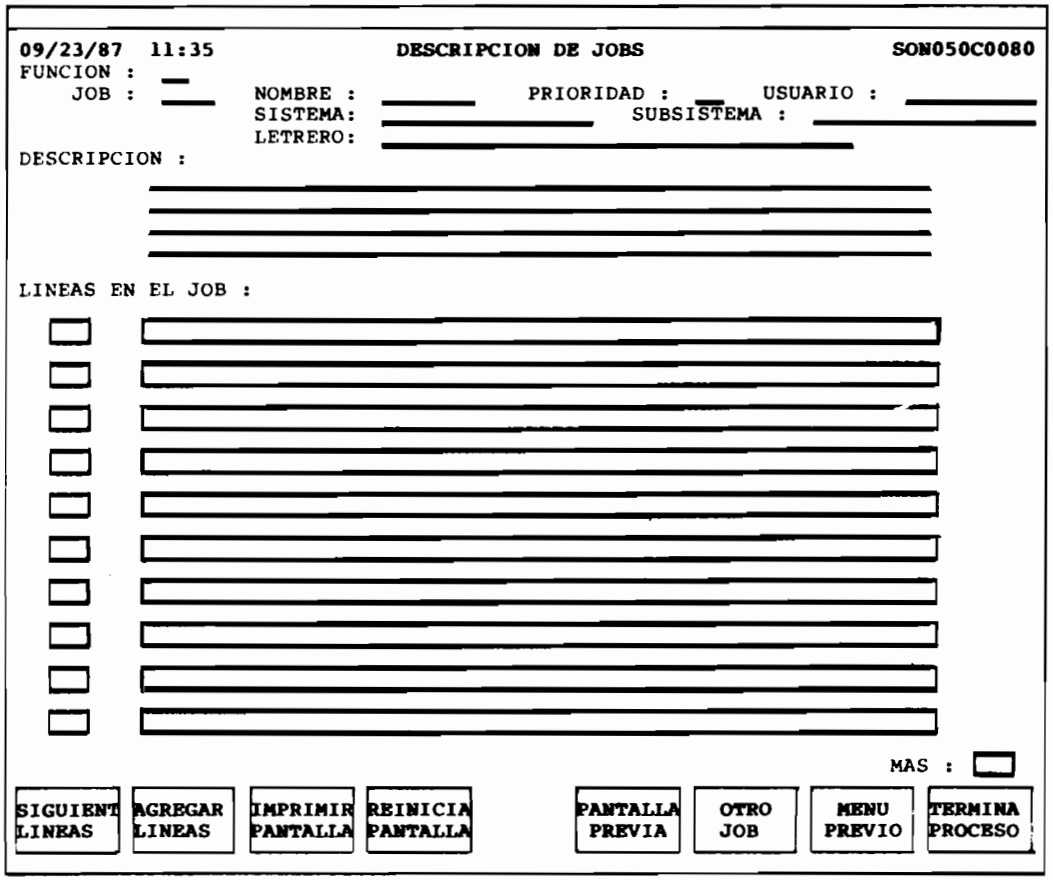
SORTED BY : System
Subsystem
Job number

CONTENTS : Job name
Label description
General description
Programs executed
Data sets affected by each program
How each data set is affected (Read/Write)

S C R E E N N O. 1


09/23/87 11:35
DESCRIPCION DE JOBS
SON050C0080
FUNCION :
JOB :
NOMBRE :
PRIORIDAD :
USUARIO :
SISTEMA:
SUBSISTEMA :
LETRERO:
DESCRIPCION :
LINEAS EN EL JOB :
MAS :
SIGUIENT REGISTRO
PRIMER REGISTRO
IMPRIMIR PANTALLA
REINICIA PANTALLA
BORRAR PANTALLA
LINEAS DEL JOB
MENU PREVIO
TERMINA PROCESO

S C R E E N N O. 2


09/23/87 11:35
DESCRIPCION DE JOBS
SON050C0080
FUNCION :
JOB :
NOMBRE :
PRIORIDAD :
USUARIO :
SISTEMA:
SUBSISTEMA :
LETRERO:
DESCRIPCION :
LINEAS EN EL JOB :
MAS :
SIGUIENT LINEAS
AGREGAR LINEAS
IMPRIMIR PANTALLA
REINICIA PANTALLA
PANTALLA PREVIA
OTRO JOB
MENU PREVIO
TERMINA PROCESO

S C R E E N N O. 3

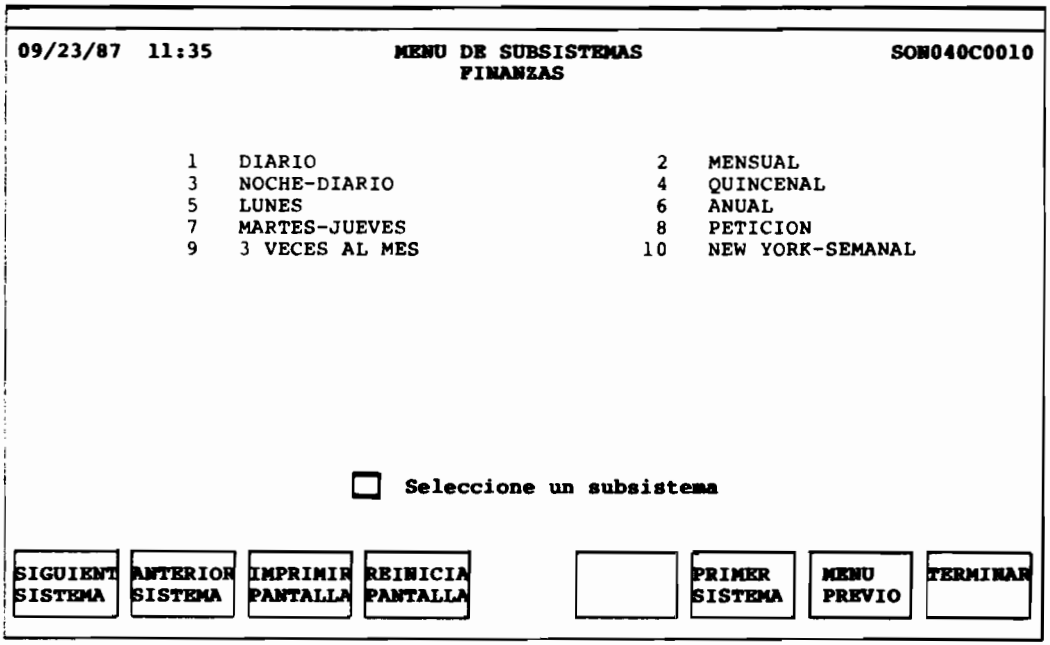
09/22/87 10:15 **MENU DE SISTEMAS** SON040C0000

1	SISTEMAS	2	PERMISOS
3	HISTORICOS	4	TELEX
5	COMPRAS	6	AUDITORIA
7	CONTROL DOCS.	8	CONSULTAS
9	FINANZAS	10	ITA

Seleccione un sistema

		IMPRIMIR PANTALLA	REINICIA PANTALLA		PRIMER SISTEMA		TERMINAR

S C R E E N N O. 4


09/23/87 11:35
MENU DE SUBSISTEMAS
FINANZAS
SON040C0010
1 DIARIO
2 MENSUAL
3 NOCHE-DIARIO
4 QUINCENAL
5 LUNES
6 ANUAL
7 MARTES-JUEVES
8 PETICION
9 3 VECES AL MES
10 NEW YORK-SEMANAL
Seleccione un subsistema
SIGUIENT SISTEMA
ANTERIOR SISTEMA
IMPRIMIR PANTALLA
REINICIA PANTALLA
PRIMER SISTEMA
MENU PREVIO
TERMINAR

S C R E E N N O. 5

09/23/87 11:35 **MENU DE JOBS** SON040C0020

FINANZAS **MENSUAL**

69	AJUSTE MENSUAL SIST CNTL PRESPSTL	71	REPORTE DE PAGOS CON AUT. PRESUP
75	REPORTE DE PASIVO	77	REPORTE ESTADISTICIO DE PAGOS
94	REPORTE DE FINANCIAMIENTOS	109	PAGOS DE COMERCIO INTERNACIONAL
111	REPORTE DE ADEFAS PAGADAS	111	SOLICITUDES DE FINAN. PENDIENTES
114	ERROR EN INFORMACION PRESUPUESTAL	130	ESTADISTICAS POR TIPO DE ORDEN
142	SUBIR A CINTA CONFIRMACIONES DE FIN		

☐ **Seleccione un job**

SIGUIENT SUBSISTE | **ANTERIOR SUBSISTE** | **IMPRIMIR PANTALLA** | **REINICIA PANTALLA** | | **PRIMER JOB** | **MENU PREVIO** | **TERMINA PROCESO**

S C R E E N N O. 6

09/23/87 11:35 PARAMETROS DEL JOB SOLICITADO SON040C0020
FINANZAS MENSUAL REPORTE DE PAGOS CON AUT. PRESUPUES

FECHA : HORA :

1 FECHA DE INICIO DEL MES

AAMMDD

2 FECHA DE FIN DEL MES

AAMMDD

3 NUMERO DE COPIAS

2

4 TIPO DE PAPEL (SI NO SE PONE ES PAPEL NORMAL)

5

6

7

SIGUIENT JOB | PARAMETR PREVIO | IMPRIMIR PANTALLA | REINICIA PANTALLA | CANCELAR JOB | EJECUTAR JOB | MENU PREVIO | TERMINA PROCESO

BASE DE DATOS DE CONTROL DE PROYECTOS Y DOCUMENTATION DBDOC

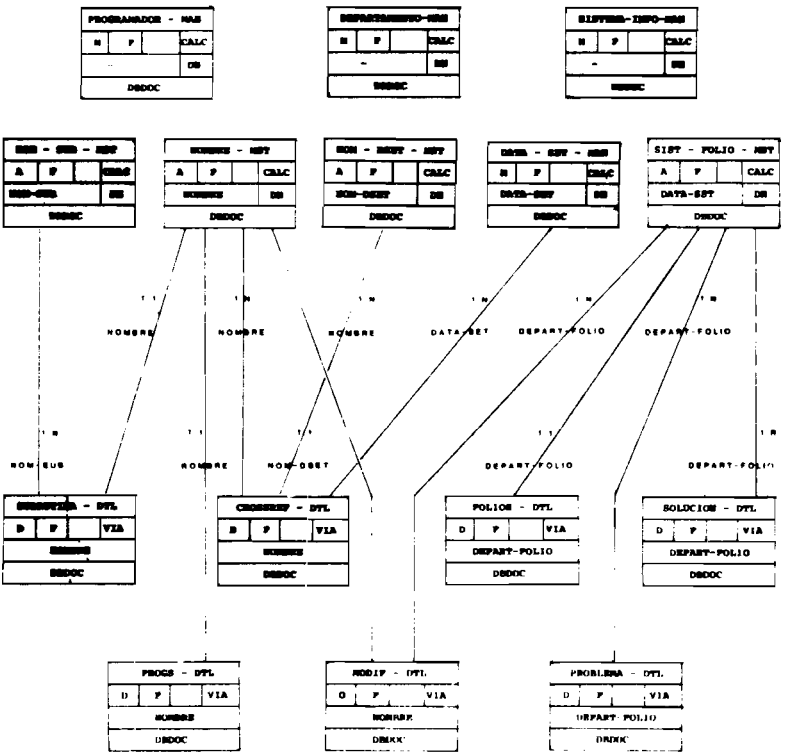


FIGURE 1

SISTEMA DE JOBS EN LA BASE DE DATOS

DBTBL

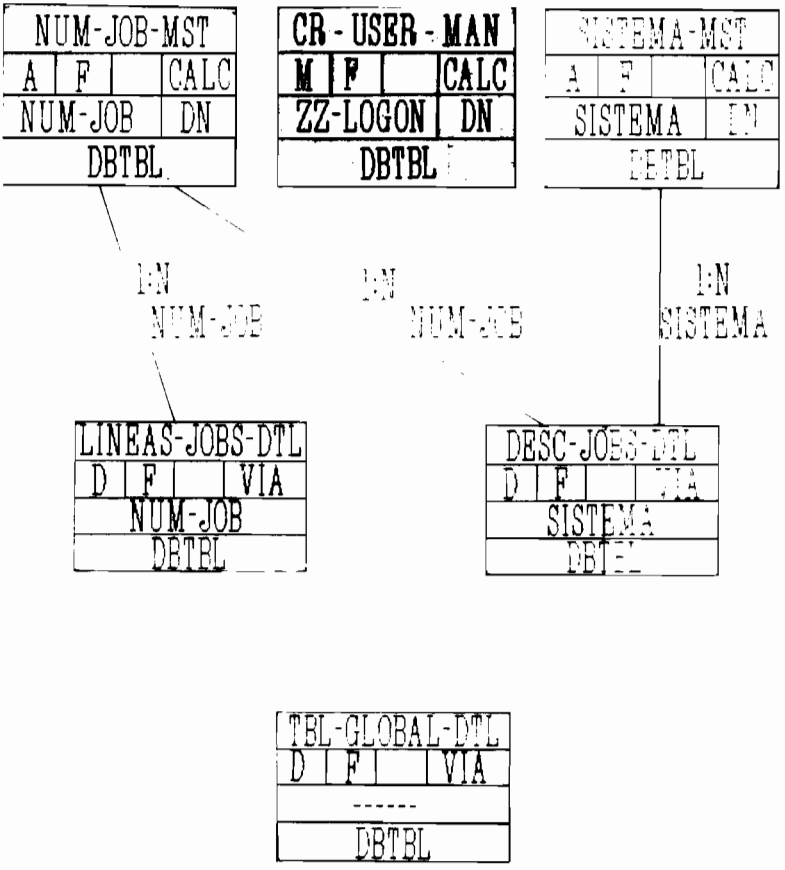


FIGURE 2

JOBS SYSTEM STRUCTURE

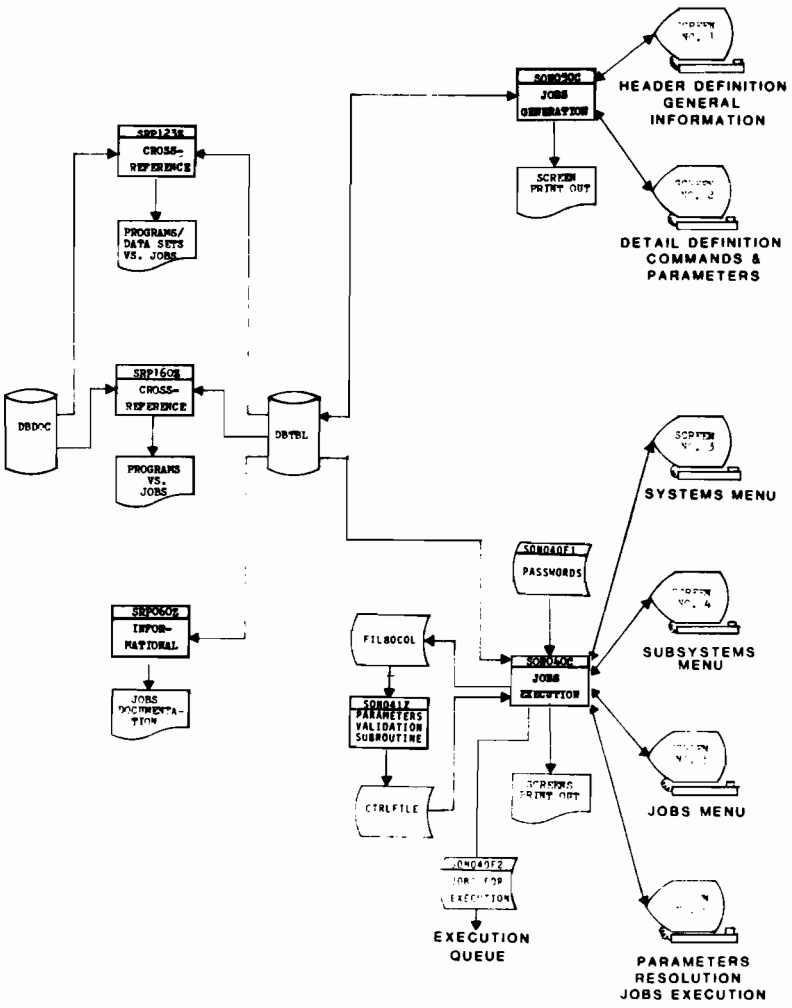


FIGURE 3

Enhancing the Functionality of HPDeskManager

Kim Sasko
Hewlett-Packard Co.
525 West Monroe, Suite 1308
Chicago, IL 60606

HPDeskManager is an office tool you can use to transport your company's information. All types of information can be distributed through HPDeskManager. By writing a very simple script file or a simple FSC program, your company can use HPDesk to distribute information such as, data base reports, programs, all types of text documents and reports generated using Basic Report Writer (BRW). Script files and FSC allow our customers to easily distribute their company's information using HPDeskManager.

The topics discussed in this paper include; identifying the types of information to be sent through HPDesk, determining the best alternative to send the information, and how to write the interface to provide this functionality to your users.

Identifying the type of information to be sent

What information does your company generate and send through the U.S. Mail or company internal mailing system? If this information is generated on an HP3000, HP 150 or Vectra, it may be easier to distribute this information using HPDesk. There are few limitations to the type of mail which can be sent in HPDesk. The limitations are that the file must reside as either an MPE file on disc or an HPDesk item and the file size must not exceed 4,000 sectors. This sector limit is associated with all messages which are to be transported through a network of HP 3000s. A message which is not intended to leave the local HP 3000 can be much larger in size.

Some examples of the types of information sent by Hewlett-Packard employees through our internal company-wide HPDesk network are:

- Financial reports*
- Manufacturing reports*
- Software patches for HP-developed software applications
- Presentation drawings developed for marketing and teaching purposes
- Product marketing surveys
- HP-Internal newsletters
- Software product updates
- Order processing invoices

* Reports can be generated using Basic Report Writer (BRW).

Determining the best alternative to send the information

Once the type of information to be sent is decided upon, answer the following questions. These questions will help you to determine the most practical way to send your information through HPDesk.

1) How many sectors of disc space is the file?

   The default configuration for the maximum item size of an HPDesk item is 20 Kbytes, approximately 80 sectors. The maximum size of a message that can be transported by HPDesk is 1024 Kbytes or 4,000 sectors. Once you decide on the maximum item size (the size of a basic item) and the composite item size (the size of the entire message) configure the sizes in the Data Menu of the HPDesk configurator. Be sure that these sizes are configured the same on all machines in your network. If the maximum item size is not configured the same on all machines, the file will only be accepted on the machines that are configured large enough to import the message into the HPDesk data base.

2) What is the record size of the file?

   HPDesk displays messages in 80 byte records. The record size of a file will determine how the file will be displayed in HPDesk. If the record size exceeds 80 bytes, the text will be truncated at 80 bytes, and the remainder of the text will be inserted on a new line. This being the case, you may decide that you do not want the file displayed in HPDesk. Instead, you may decide to transport it as an MPE file. Transporting MPE files works the same as any other message, except that the text portion of the message is not displayed. The MPE file can be copied out of HPDesk to an MPE file and used accordingly. HPDesk does not alter the contents or format of MPE files when it transports them through the network.

3) Will the information be sent through HPDesk for display purposes only, or will it need to be accessed by another subsystem other than HPDesk?

   If the file will be accessed by another subsystem other than HPDesk, you may want to transport the file as an MPE file. When an MPE file is copied into HPDesk as text, the file becomes a "text" file. This conversion removes the tab markers and other format characters from the file. Altering the the file, by removing format characters, may alter the way the file is formatted by the other subsystem. The file can be sent both ways, as a readable "text" file and as an MPE file.

4) Will the information be sent on a regular basis? If so, will the information have the same MPE file name each time?

   If the information is referred to by the same file name, or similar file name, commands to retrieve the file from MPE can be set up in HPDesk. Commands to do this type of operation can be installed in HPDesk and are referred to as "script" files. Script files can be set up to execute a series of commands. These commands can check for the file, and if present, send it to a specified distribution list of people -all by issuing one HPDesk command.

5) Will the information need to be shared with foreign electronic mail users?

   If the file is to be shared with foreign electronic mail users or other subsystems on foreign systems, you can take advantage of HPDesk's foreign service connection.

6) Would your company benefit by having programmatic access to HPDesk for the purpose of sending files?

   It may be more convenient for your company to write an FSC application to automatically import MPE files into HPDesk. This type of interface would not require any user intervention. The interface would require that the files being transported be converted to ASCII format.

Alternatives for transporting the files

Messages can be sent through HPDesk using various types of interfaces in HPDesk. The three alternatives described below include; interactive, script files and FSC.

Interactive

Manually sending various types of information is the most common way to send mail through HPDesk. If the file resides in MPE, the file can be sent by using the following commands;

```
In Tray >  send (filename)
    TO:    John Smith
    CC:    Mike Doe

Message >  acknowledge 4
Message >  mail
```

The manual interactive method works best for ad-hoc reports and other types of information which is not sent on a regular basis.

You may want to use HPDesk to view the information you will sending before you determine exactly how you want to send it. Copying the file into HPDesk and then reading that item will show you how the information will be viewed by the HPDesk users who receive it. The commands to do this are:

```
Copy text from (file name)
Read it
```

Script files

HPDesk B provides our customers with a facility to write their own compound set of MPE and HPDesk commands, called scripts. Scripts allow our customers the flexibility to write their own commands for HPDesk, install the commands in HPDesk and provide these commands to their users as enhanced functionality to the standard HPDesk product.

The process of creating and installing a script file consists of the following:

1) Create the commands for the script file as either a text item in the work area of HPDesk or as an ASCII file in MPE using Editor, TDP, etc.

2) Once the script is written, it can be provided for use by other users by installing it in the Applications Area of HPDesk. To do this, go to Area 11 in HPDesk, type install script, and answer the prompts. You must have user capability of Script Administrator to install the script. The last prompt will ask which group of users should have access to the script. You can choose the group, Everyone, or any specific user group defined by your HPDesk administrator.

3) Once the script is installed, users can access the script by typing the name of the script, as if it were any other HPDesk command.

Setting up script files works very well if you want to send reports which are generated on a regular basis or other types of information which are sent to a specified distribution list of people regularly.

Examples of script files can be found in the Programmatic Access to HPDeskManager Reference Manual. Some modified examples from the book are below. The three commands below send a designated MPE file to an existing distribution list, set the acknowledgement level on the message to READ and then MAIL the message.

```
send (file name) to "distribution list"
acknowledge 4
mail
```

If the file is a non-text file it will be sent as an MPE file and will be non-displayable by HPDesk. To make the file displayable by HPDesk, the following script could be used:

```
copy text from (file name)
send it to "distribution list"
acknowledge 4
mail
```

To verify that the file actually exists before sending it, you could use the following commands:

```
&forward itemok <exists <file name>>
&print The file does not exists
&exit
&itemok Continue to send the item
copy text from (file name)
send it to "distribution list"
acknowledge 4
mail
&exit
```

Foreign Service Connection

Foreign Service Connection provides programmatic access to HPDesk. To send a file to a foreign electronic mail system, FSC can be used to output the messages to MPE for easy access by an external application. The external application would be responsible for transporting the file to the foreign system and getting it imported into the foreign electronic mail system. FSC also requires a foreign service gateway to be configured in the HPDesk configurator.

Programmatically importing messages into HPDesk can be accomplished using FSC. The application is very straight forward to write. You can read about all of the details related to writing an FSC application in the Programmatic Access To HPDeskManager Reference Manual. Actual file formats for messages, etc, are well defined in this manual. The major points necessary in writing this type of application are described below.

Brief overview of the program specifications:

1) Names of MPE files to be transported must be identified.

2) Standard header information for each message must be provided. The header information consists of:
   - Sender's name
   - From names
   - To names
   - CC names
   - BCC names

- Subject of the message
- Date
- Optional ID, acknowledgement, and priority settings.

3) The MPE file must be formatted in ASCII and appended to the same file containing its header information. Note that non-text items cannot be sent programmatically using FSC.

4) An IPC file must exist in the MAILDB group of the HPOFFICE account by the file name ARPAIPC. This file is used to keep record of all MPE files which have been formatted for the FSC application to transport into HPDesk.

5) The FSC application, or your operations staff, must be sure to enable the FSC ARPA Truck. This truck is responsible for reading the ARPAIPC file, converting the ARPA formatted files into HPDesk binary format, and importing the files into HPDesk.

Actual file formats of the IPC files, and ARPA files is included in the Reference Manual. An FSC application written to format and import messages can be quite simple. There are some things to look out for. Non-text messages have already been discussed as being unacceptable for FSC. Record size is another factor to be considered. FSC supports 80 byte records. If your files are less or equal to 80 byte records, formatting the file will be quite simple. If the record size exceeds 80 bytes, the file will need to be reformatted into 80 byte records for FSC. The most complex part of the FSC application, in the case, is the reformatting of the text. Examples of each of the components, including an actual application to generate an ARPA message for import into HPDesk, is included in the reference manual.

Writing FSC applications to connect HPDesk to foreign electronic mail systems is more complicated than the application described above. Many of the complications are related to compatibilities between the products. The FSC application not only has to format the message for incoming mail, it must also pick up the outgoing mail and make sure it transferred to the foreign system in its entirety. Compatibilities such as directory names, addressing sequences, and file formats must all be considered before writing the application. These topics will not covered in this paper due to the level of detail necessary to fully explain the specifications for a program(s) of this kind.

Summary

In summary, HPDesk is a very versatile product. With the availability of script files, the functionality a company can now provide to its users is quite challenging. Script files will now allow our customers to enhance HPDesk to the level of functionality they desire. In the case of transporting company information, scripts can help simplify the procedural steps required to send some types of information. The time savings which can be experienced by electronically mailing information is considerable. If you depend on HPDesk for messaging today, writing enhancements to increase the flexibility and usability of the product is quite a benefit.

ASSESSING YOUR OFFICE AUTOMATION NEEDS

Kim Sasko
Hewlett-Packard Co.
525 West Monroe, Suite 1308
Chicago, IL 60606

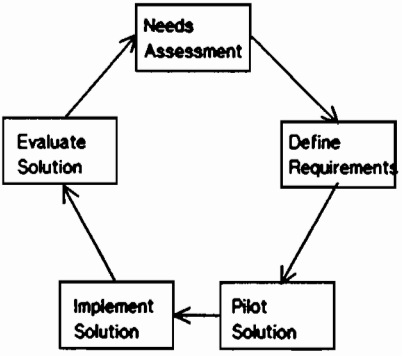
Understanding the needs of office users is important to the successful selection, pilot and implementation of office solutions. If you understand the bottom line needs of your users you can make them more productive. This paper discusses the topic of assessing your office automation needs. Throughout this article I will be discussing what a Needs Analysis is, why it is important, how to conduct a Needs Analysis and how to compile the results of the analysis.

Office Automation provides continually changing technology to its users. Six years ago office automation was Word Processing. As office users began to share information and become better at manipulating the information available to them, office solutions evolved into information distribution (electronic mail) and information processing (list processing, spreadsheets, etc.). Today, office automation is the integration of all office information.

The buyers of office automation solutions must determine which office tools will best fit their users' needs and best help their company achieve its business goals. Many office solutions look good, but if the solution is not going to increase the productivity of your users and therefore help your company accomplish its goals, it may not be the right solution for you.

What is a Needs Analysis?

A Needs Analysis can be defined as the process used to identify the functionality necessary for a user or group of users to perform their job efficiently and effectively. These functional needs are identified as functions only; they are not specific products, solutions or features. The diagram below illustrates where the Needs Analysis process exists in the cycle of selecting, implementing and maintaining office products.



Surveys, interviews, and observations can all be used as input for identifying users' needs in an office environment. A department within your MIS organization should be made responsible for this activity. The Needs Analysis also requires cooperation of your current or potential office users. Knowledge of your company's management style and corporate objectives is necessary. Knowledge of office automation technology is also necessary.

The Importance of Performing a Needs Analysis

Being involved in the pilot or implementation phases of an office product can be an extremely challenging experience. If you have actively participated in this type of activity, you can probably remember at least one surprise that may have arisen during the pilot phase. While an unexpected problem may not cause a pilot to fail, too many bad experiences can make a pilot a very unpleasant experience. Keeping the surprises to a minimum is one benefit to assessing your office automation needs before implementing the solution.

Unexpected problems are not only related to pilots. As more and more office users become computer knowledgeable, their expectations change, they become more sophisticated. As a result of this "sophistication" a number of unknown circumstances could arise. A product characteristic such as slow terminal response time, can be cause for serious concern and dissatisfaction by the users. Other aspects of a product, such as number of keystrokes or non-user friendly interfaces, can also be easily criticized. Products with these characteristics can be successfully implemented. Understanding the users' expectations, current environment, and functional needs will help make new office automation solutions successful.

Many other benefits can be achieved as the result of performing a Needs Analysis. Not all companies will experience the same benefits. And within a company, some departments may achieve better results than others. To a certain extent, your success may depend on the level of functionality already automated successfully. The type of relationship your MIS organization currently maintains with its user base also has some impact on the results. Some of the benefits which can be achieved by performing a Needs Analysis are described below.

1) Unknown needs may be identified:

   Asking users how they perform various functions of their job, will assist you in finding out about several specific tasks the users perform. Learning of these tasks will help you identify office needs that may have been unnoticed in the past. In addition, having surveyed the users, selected, piloted and implemented a product does not mean that the product is a perfect fit. Nor does it mean that the user is now automated as efficiently and effectively as possible. The process of implementing office solutions is a cycle. Users' needs change and as the availability of office

solutions continues to increase, users demands for better solutions also increases.

2) Existing products/technologies may be identified as being underutilized:

   In asking the users questions about how they perform certain tasks, you may find out that some incorrect assumptions had been made. For example, you may have only trained the users on how to send certain types of documents through electronic mail. In talking with the users, you may find that they need to send a wide variety of document types, including PC files, through electronic mail. In this case, the lack of functionality was not the result of the product, rather a misunderstanding of how a user needed to use the product.

3) Better assess the current level of office productivity being achieved:

   A Needs Analysis can provide you with insight on the extent to which your users are using office automation solutions. This can be done by performing an "inventory control" of your current office environment. Surveys providing a list of hardware and software products installed can be used to collect this information. As a result of the survey, you can find out just how heavily some products are being used.

   In addition, given the right tools and training, users can find very innovative ways to use products. Lack of functionality in a product can be made up by user innovativeness. The MIS department should be aware of user-developed solutions, since these solutions are filling a functional need required by users.

4) Assist in justifying the cost of future OA solutions:

   Justifying the cost of future OA products is simplified if you can measure actual productivity gains. For example, you may already know that 40% of a "non-automated" secretary's day is spent typing. Based on the results of automating other secretaries, you found that the time spent typing was reduced by 25%. So, by having already automated certain functions, productivity gains are so apparent that increasing the level of automation will increase user productivity even more.

5) Build users' confidence in your knowledge and ability to support OA:

   Showing interest in what the user is doing, and perhaps sharing some of your plans for the future, helps you build a good working relationship with your users. This relationship is important when you begin to involve the users in the pilot, implementation and evaluation phases of OA solutions.

6) Become familiar with the attitudes various users and departments have towards OA and change in general:

   If you are looking to select products for your users, it is beneficial to understand users' attitudes. Once a product is selected for a pilot test, carefully choosing users for the pilot group is important. Users who are resistant to change or do not look forward to improving their work environment will probably not contribute to the success of a pilot.

7) Develop a clearer definition of your company's OA requirements for the development of a Request for Proposal:

   If, ultimately, the purpose of performing the Needs Analysis is to purchase an OA solution, the development of an RFP will be simplified. Once the results have been analyzed, the users needs can be outlined for inclusion into an RFP document.

How to Perform a Needs Analysis

The process of performing a needs analysis consists of several steps ranging from surveying the users to presenting the results to management.

Prepare Surveys and Interviews
Distribute Surveys/Perform Interviews
Analyze Results
Document Constraints
Present Results to Management
Prepare Functional Needs Matrix
Prepare RFP
Work with Vendors to Select Solution

This diagram outlines the list of steps necessary to perform a Needs Analysis. Each of the topics listed are discussed in the later sections of this paper. If you are not performing the Needs Analysis for the purpose of purchasing a solution, not all of these steps are required.

PREPARE SURVEYS AND INTERVIEWS

Carefully preparing the surveys will assist you in gathering the information necessary to identify the functionality needed by your office automation users. There are several types of surveys which you can use. Some survey questions may only be relevant to certain types of users. Surveys can be organized according to the following groups of people; management, professionals, administrative support and word processing operators.

Each survey should include several questions which are categorized according to functions performed by the people in your office. Described below are example categories and questions.

1) User Expectations:

The questions in this section ask the users what their expectations are about using office automation products. When looking at office automation needs, understanding the users' expectations may be very helpful when it is time to select a product. Even a product which may appear to be a perfect fit on paper can fail if it does not meet certain expectations which have already been determined by the users. Below are some questions which may pertain to your computer users. Not all of your users will understand these questions, so use whatever method is most understandable to your users to find out this information.

a) What is acceptable terminal response time?
b) What is acceptable print quality?
c) Is impact printing a printer requirement?
d) What is acceptable print speed?
e) Do ergonomics play an important role in the hardware selection process?

2) Current Environment

The questions in this category help identify how the job is getting done today. This type of survey may also uncover what is working well for your users and what is not. Understanding these points will simplify identifying and prioritizing the users needs. The example questions in this section are grouped according to the following functions; information processing, communications, document creation, graphics, mailing/filing, travel, product inventory, and information integration.

Information Processing

a) Describe the major functions carried out by this department.
b) Describe briefly, the major steps performed in the department to carry out this function. Indicate people responsible for the functions.
c) List any problems or obstacles to the performance of these functions.
d) What is needed to eliminate these problems?
e) Describe below what information is required to be accessed and/or produced to carry out the functions of your job.

-	Information Required	Source	Form	Frequency
-	Information Produced	Source	Form	Frequency

f) How do you use the information you produce?
g) What changes would you make to improve the operation of your department?
h) Describe any specific measures of efficiency in the department, such as revenue per sales representative or policies per underwriter.

i) What are the major responsibilities of your job?
j) How much time do you spend accessing/producing information each day?

Communications

a) How much time do you spend in meetings each week?
b) What is the average length of the meetings you attend?
c) How are meeting agendas and minutes generated?
d) How are meeting agendas and minutes distributed?
e) How many incoming telephone calls do you receive in a day?
f) How many outgoing telephone calls do you make in a day?
g) How much time do you spend on the telephone each day?
h) How much of your time is spent communicating with people within your company? (telephone, meetings, etc.)
i) What changes would you make to improve the way you communicate with the people in your company?

Document Generation

a) Do you type?
b) What is the primary method you use to compose written materials?
c) What kinds of paperwork do you generate and process?

	Type	Frequency	Size	# of Revs	Time
Reports					
Gen Corresp					
Form Letters					
Legal					
Financial					
Other					

d) Do you require any of the following functions:

Alpha/numeric sort	Info from computer printout
Spreadsheets	Data Base inquiry
Calendar/tickler	Math calculations
Graphics	

e) Do you use form letters?
If yes, in what quantity and for what type of application?
f) Do you use a word processing center or remote WP terminal?
If yes, for what type of work?
g) Will you have typing requirements which you have not requested in the past because the typing staff lacked the time or you considered it a low priority?

Type of document	Length	Frequency

h) How much time do spend creating/editing documents in a day?
i) What changes would you make to improve the way you handle document generation at your company?

Graphics

a) Do your documents include both graphics and text?
   If yes, please explain the type of graphics (bar charts, flow diagrams, technical drawings, etc.)
b) How are these graphics currently prepared and produced?
c) Do you produce graphics from computer files?
d) Is there a need to access a data base to produce graphics?
d) Do you produce documents which are eventually typeset?
e) Do you use graphics when making presentations or conducting meetings?
f) How much time do spend creating/editing graphic materials in a day?
g) What changes would you make to improve the way graphics are generated and used in your company?

Mail/Filing

a) Do you utilize a centralized filing system in your area?
   If yes, please describe.
b) Do you use an electronic filing system?
c) Are you satisfied with your filing system?
   Comments.
d) How much time do you spend filing materials in a day?
e) Does your secretary screen your mail, attach backup material if necessary, catalog or keep an action file?
f) Is the internal mail system adequate for your needs?
   Explain.
g) Do you use electronic mail?
h) What changes would you make to the way your company the mailing and filing of items?

Travel

a) Is travel required for your job?
   If yes, describe the nature of your trips.
b) How many days per month do you travel?
c) How do you handle communication with your office when you are away? (telephone, mail, telex, etc.)
d) What kind of information do you need to access when you are traveling (pricing, ordering, spreadsheets, documents, etc.)?
e) What kind of information do you need to produce when you are traveling (pricing, ordering, spreadsheets, documents, etc.)?

Product Inventory

Software:	Yes	No	Product Name(s)
a) Do you use electronic mail?			
b) Do you use word processing?			
c) Do you use electronic filing?			
d) Do you use spreadsheets?			

e) Do you use data base inquiry?
f) Do you use report writing?
g) Do you use electronic calendaring?
h) Do you use graphics?
i) List the products you do not currently use but would like to have made available to you.

In addition to finding out if users use certain functions, and which ones, you may want to find out the following: how much do they use each product, do they feel they are adequately trained on the products, are there specific features missing in any of the products they use, are there other products they would prefer to be using.

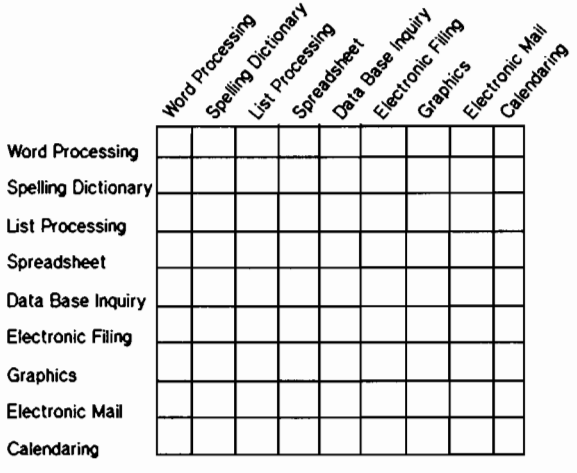
Hardware: Yes No Device Name %Used

a) Do you use a terminal?
b) Do you use a Personal Computer? If so, what kind of disc and printer do you use?
c) Do you use a Portable Computer? If so, what kind of disc and printer do you use?

In this section you are asking what hardware do you use and how much do you use it. You may also want to ask if they like what they are using, if there are other products they would prefer to use, and if there are specific features missing in any of the products.

Integration of Information

a) Using the matrix below, specify the products which you have a need to integrate.



b) If you use a combination of terminals, PCs and/or portables, specify the types of information you need to transfer between devices.
   Terminal and PC data?
   Terminal and Portable data?
   PC and Portable data?
c) Specify the types of data, for each question you answered "yes" to above. (i.e, word processing, data base, spreadsheets, etc.)
d) Is there information you would like to share/transfer between these devices, but are unable to do so at this time?
   Please explain.
e) What changes would you make to way you use terminals, PCs and portables today?

DISTRIBUTE SURVEYS/PERFORM INTERVIEWS

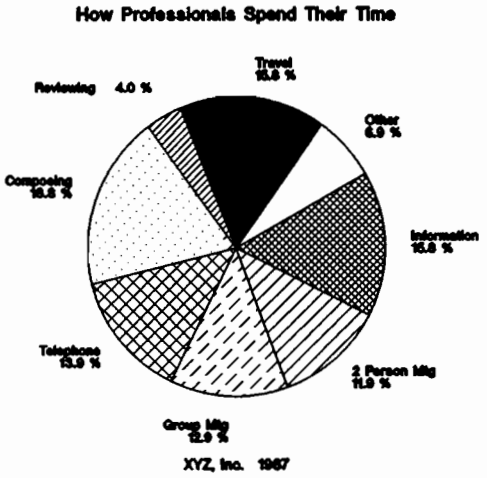
Determining which users to survey depends on your reason for performing the Needs Analysis. You may want to review a certain department or only a certain type of user, such as managers. Prepare the survey for the people you have selected. Giving management a survey on how they handle their word processing workload would probably be inappropriate. And, it is not always necessary to survey every user. However, choose several users from each functional group, or as many different functional groups as possible. Once the surveys have been distributed, make your team available for questions and comments, and encourage your users to answer the survey carefully and honestly.

When selecting interviewees, choose a cross-section of users from the various functional groups. Also, remember to choose several different management people. Future office solutions will require management commitment at all levels of the office cycle (pilot, implementation, evaluation) and you must keep them informed of the process you are using to collect information. When you present your results to management, they will be familiar with some of the tactics used to collect the data, and they will have been active participants in that part of the Needs Analysis process.

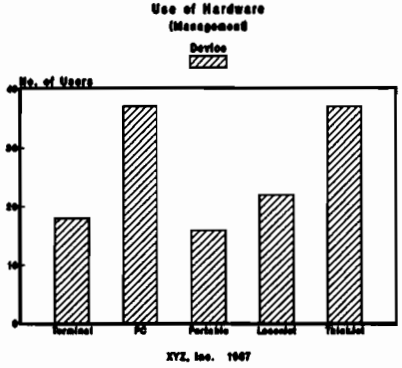
Surveys vs. Interviews. Surveys are useful in collecting quantitative information. Questions requiring yes/no answers or short answers also work well with surveys. While interviews and surveys can be interchangeable, interviews can be extremely helpful in certain areas. If used for interviewing a manager, it allows the manager the opportunity to experience the Needs Analysis process which is being performed. This may be helpful when you present recommendations later in the process. Interviews also give management, and other interviewees, time to ask further questions and/or further qualify answers or concerns.

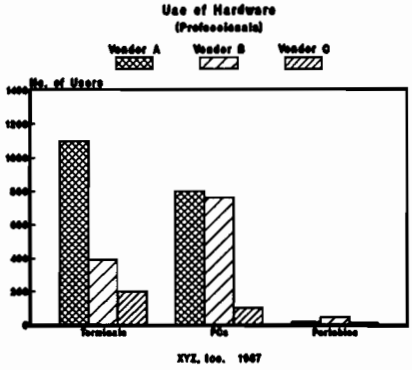
ANALYZE THE RESULTS

Summarize the results in a way that will make them meaningful to management. Once you have gathered all of the information, you could possibly use graphics to illustrate quantitative results. For example, use bar charts to show comparisons of how much certain products are being used. Pie charts represent percentage data well. For example, the chart below illustrates how professionals spend their time.



Rather than talking about the results, present them in graphical form, to allow the audience to become even more involved in your results. This type of presentation may also help the audience draw their own conclusions much faster and easier. For example, what kind of conclusions can you make about the two drawings below?

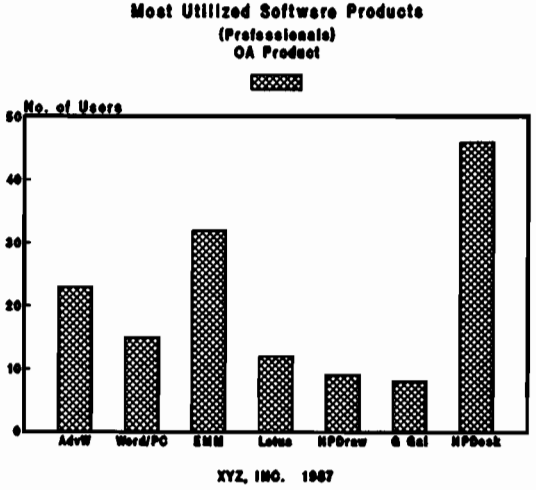




It could be cumbersome for the listener to understand these results. And unless the listener is taking good notes, the data could cause unnecessary confusion. Not only does the listener suffer; the presenter could make a much more impressive point by displaying one picture, rather than going into a five minute discussion to make the same point.

Presenting qualitative data works very well in a verbal discussion. Inflection and volume in your voice can be used to make a strong point come across in a meaningful manner. Qualitative data is also much easier to discuss than a long list of numerical results.

Analyzing the completed surveys and interviews requires summarizing the data collected, comparing the summaries where appropriate and looking for patterns. These activities should provide you with enough information to draw useful conclusions and make any necessary recommendations. Comparing the results should give you an indication of how relevant the data is to certain types of users or functional groups of users. For example, the diagram below may clarify the need for certain software products.



Looking at how each user responded to certain questions can indicate how useful a product is to the users and how well it fits their needs. Were there a lot of complaints about a particular product or feature? If so, maybe that product should be replaced with another alternative product. How do people communicate? Is it cumbersome for users because they need to share information, but cannot do so easily? If so, you may want to propose a new configuration which allows easier access to shared data.

DOCUMENT CONSTRAINTS

If there are constraints which could affect your selection of future office solutions, document them. Also think about documenting any assumptions which are being made and have some baring on your conclusions and/or recommendations. Some examples of constraints and assumptions which may apply to an office environment are as follows:

Constraints

a) All electronic mail packages must provide connectivity to the IBM PROFS product.
b) All word processors must provide conversion to the DCA document format standard.
c) All portable computers must be the HP Portable Plus.

Assumptions

a) The 1,100 PCs already in place today, will not be replaced.
b) Graphics is only used by the Graphics and Arts Department.
c) Dial-in capability to the HP3000.

Documenting constraints and assumptions provides some formal ground rules understood by all. If, after looking at the results of the surveys, it makes sense to change or eliminate one of these factors, it can knowingly be done.

PRESENT RESULTS TO MANAGEMENT

Using the information collected and analyzed above, you should now be in a position to present your findings to management. Be prepared to not only have results, but also the data to substantiate your results. Be knowledgeable of the survey questions, any particularly interesting results, and any potential problems. Discuss some particular problems some of your users are experiencing and how these problems could be resolved.

Once you can provide your management with a good understanding of how things are today, you then move on to some possibilities for the future. Be prepared to make some recommendations based on your findings. Remember, that you have survey results as substantiating data if needed. If a questions comes about how much a particular product is used or liked, graphical representation is a good way to re-summarize the results and possibly show any necessary comparisons.

Summary

Some form of a Needs Analysis can provide your company with a methodical approach to determining your users' needs for office automation solutions. The newest technology is not necessarily the best if it is not going to help your company achieve its business goals. Using surveys and interviews to gather data will provide you with some information on the current level of productivity being achieved by your users today. This same information can also help identify what office tools users feel is necessary for them to perform their job efficiently and effectively. Having gathered and presented these results to management, a decision can be made on whether to progress to the next step; inviting vendors to propose solutions which fit the requirements defined. At this point, the vendor is able to look at the problems your company is facing with automating your office users and make recommendations based on their knowledge and experience of your company and the office automation industry.

ABSTRACT

The Lion and the Mouse Revisited

Robert V. Scavullo, Noesis Computing Company

In 1984, at Interex Montreal, I first spoke about how the PC, (the mouse) was able to assist the HP-3000 (the lion). Now, in 1987, it isn't clear which computer is the mouse and which is the lion.

Last year we equipped all of our programmers with PC workstations. This year, several programmers have portable PC workstations to carry their "tool kits" with them when they work from home or on other HP-3000's. Part of my talk will describe our "tool kit" and the productivity improvements we've seen from using it.

This year, for the first time in designing new multi-user database applications, we've had to seriously weigh working on the HP-3000 or building a local area network solution. HP's recent introduction of resource sharing which allows an HP-3000 to act as the LAN file server makes the HP-3000/LAN decision even more interesting. The second part of my talk will cover our rules of thumb for allocating applications to the HP-3000 or an LAN.

A "Message" About [Turbo]IMAGE

by

Gilles Schipper
G. Schipper & Associates Inc.
P.O. Box 310
Thornhill, Ontario L3T 4A2
Canada

ABSTRACT

Many successful HP3000 commercial business application systems are built upon the solid foundation of the [Turbo]IMAGE data base management facility available with every HP3000 system.

Typically, the most complex programs associated with DBMS-based systems are those that are interactive, and also involve several - even many - on-line users accessing the same data base(s). Furthermore, a very large majority of these programs are designed to utilize IMAGE open mode 1- which provides for shared modify capability - but is also the most difficult to design and program, and consumes the most system resources, thus executing least efficiently.

There exists an alternate design strategy, involving the use of probably the most useful file management facility introduced to the HP3000 family of computers since IMAGE- and also one of the best-kept secrets in the HP3000 world- namely, the Message-File feature of the Inter-Program Communication (IPC) facility.

With IPC, on-line programs used by multiple on-line users, accessing one or more [Turbo]IMAGE data bases (as well as other files, such as KSAM) are easier to program, run much faster, and offer significant additional benefits - such as transparent data base backup while users continue to access the data base.

This paper will describe the significant benefits associated with IMAGE/IPC design, together with associated techniques and caveats.

INTRODUCTION

The Interprocess Communication Facility (IPC) is a subset of the MPE File System, introduced to the family of HP3000 computers in late 1981, as part of the MPE IV operating

system.

IPC, though not highly publicized (a scant 3 pages in the Communicator corresponding to MPE IV), was a blessing to application system designers and programmers alike, who wanted an simple, efficient, and NON-PRIVILEGED method of enabling related (or non-related) processes to communicate with each other.

Before IPC, designers wanting this capability were forced to trade off ease of implementation against run-time efficiencies - utilizing IMAGE, KSAM, or flat MPE files for relatively simple application construction (and poor execution performance) - or extra data segments (DS), multiple RIN (MR), and even privileged mode (PM) capabilities, which required very delicate and sophisticated programming abilities to gain significant execution performance advantages.

In most cases, however, systems were designed to include all transaction processing requirements in a single process, without even considering the benefits of "distributed transaction processing" from an application design point of view - simply because the techniques available at that time created more problems than they solved.

Even today, fully six years after the introduction of IPC, system analysts have been slow to incorporate this outstanding feature of the MPE File System into their application designs.

INTERPROCESS COMMUNICATION FACILITY (IPC)

Simply stated, the Interprocess Communication Facility (IPC) permits multiple user processes to communicate with each other in an easy and efficient manner.

This is achieved by the use of a special kind of MPE file, known as a message file (or memory file), which acts as a first-in, first-out (FIFO) queue of records. These records are placed into the file by FWRITE's, and removed by FREAD's. Implementation is as simple as building the file (with the ;MSG parameter option on the :BUILD command), and programmatically issuing appropriate READ's and WRITE's, as with normal MPE flat files. Additional features (such as non-destructive reads, extended waits, etc.) are easily accommodated with calls to FCONTROL - and are fully described later.

Typically, message files are associated with multiple writers and one reader, although any combinations are

permissable.

One very attractive feature of message files is its potential for reduced I/O overhead. If writer processes and reader processes are synchronized, all I/O generated by reads and writes on the file actually take place in memory (hence, the label "memory file") instead of on disc, which is many magnitudes of times slower.

TRADITIONAL DATA BASE APPLICATION DESIGN

The typical on-line application on the HP3000 today, utilizes IMAGE data bases in shared modify environment. This allows multiple users to read, add, update, and delete data entries in real-time to provide up-to-date information for all users of the data base, at any point in time.

This is really the essence of modern data processing technology, when compared to the data processing state-of-the-art of a generation ago, which was characterized by batch processing and long delays between data entry and "data base" updates, so that data was never really "up-to-date".

On-line, shared modify programs must open (one or more) IMAGE data base(s) in mode 1. Consequently, all modifications to the data base(s) must be accompanied by appropriate calls to DBLOCK - which, in turn, tend to reduce transaction throughput because of associated overheads and delays involved - not to mention the additional design and programming complexities due to the required thorough and delicate implementation of a proper locking strategy.

By their very nature, on-line, terminal I/O intensive, shared data base update programs are large and complex. A design philosophy which results in the simplification of on-line programs, as well as minimizing conflict resolution overhead (locking) can only serve to improve the technology of on-line transaction processing.

Enter the marriage of IMAGE and IPC technology.

A NEW APPROACH TO ON-LINE TRANSACTION PROCESSING

Instead of opening the data base in shared modify mode (mode 1), open it instead with mode 6 (shared, read-only).

Obviously, users would no longer be DIRECTLY modifying the data base in this mode. Rather, the on-line program would be outputting all add, update, and delete transactions to a

message file.

Concurrently, a batch program (no, we are not reverting to a previous generation) is actually responsible for performing the corresponding DBPUT, DBUPDATE and DBDELETE operations on the data base. The batch process needs to open the data base in mode 4 (exclusive update, allow concurrent multiple mode 6 readers).

The resulting inherent delay between transaction generation and transaction completion, is obviously volume dependant- yet, in most cases, surprisingly short.

Consider the advantages:

(1) Program complexity is distributed. Rather than have your on-line, terminal I/O intensive programs also be responsible for data base modifications and conflict coordination, this task is off-loaded to a typically far less complex batch program, which is generally also much easier to debug and maintain.

(2) Overall program complexity is significantly reduced. The on-line program requires no locking (with some exceptions, described later). The batch transaction processor program requires no locking since it has exclusive update access to the data base. (Again, exceptions are described later)

(3) Execution performance is substantially improved. Mode 4 modifications entail less overhead than mode 1 modifications since contention conflicts (normally implemented with locking) need not be dealt with.

(4) Terminal response times are superior, since on-line users do not wait for data base I/O to complete. Instead, all major I/O processing is delegated to the batch transaction processor, which can easily be outfitted with a dynamic, self-tuning capability, to respond to varying overall data processing conditions, and speed up or slow down its own execution. More on this technique later.

(5) Data base backups can be performed, in a manner which is virtually transparent to the on-line users.

(6) Problem resolution is simplified and can be far less time-consuming - and, in many cases, can be undertaken without inconveniencing on-line users.

MECHANICS AND TECHNIQUES

Despite the general overall ease of implementing this design

approach, it still requires some sophistication, care, and ingenuity - particularly in dealing with some new problems introduced with this new design philosophy.

Let's identify some potential pitfalls, and appropriate remedies, together with some useful techniques to realize the full potential of this methodology:

1. Undesired batch process termination.

In its default condition, a message file reader process will suspend (wait) upon its **FIRST** message file read even with no writer processes (on-line terminal I/O programs) active. Subsequently, the writer process will suspend if there are no records available for processing **AND** there is at least one active writer process. In the event that all writer processes have completed (say, lunch time), the reader process will reach an "end-of-file" condition, and, presumably, terminate.

To avoid this situation, and eliminate the requirement to repeatedly re-launch the batch processor, two options are available. First, a call to "FCONTROL" with controlcode equal to 45, immediately following the "FOPEN", enables "extended wait" on the message file reader, and prevents a physical end-of-file condition from being reached. Alternatively, though less desirably, it's possible to simply close then re-open the same file upon reaching end-of-file condition. In either case, termination of the batch process is achieved in an orderly and pre-determined manner, by creating a very simple FCOPY job stream which places a "logical" end-of-file trigger record (say, with "99" in the first 2 characters) which is appropriately recognized by the batch processor program.

2. Illogical transactions (such as duplicate adds, duplicate deletes, updates to supposedly deleted records, etc.)

Since there is at least some delay between the generation of a transaction from the writer process and the final placement of that transaction into the data base by the reader process, how does one preserve the logical integrity of the application in those situations where an attempt is made to add the same record twice, or an update is attempted on a record that logically does not exist, but physically does - to name only two examples?

The solution to this problem is to provide a separate data base whose one role, among others, is to maintain transaction-in-progress records which are continually

updated (by both the reader and writer processes) to reflect the logical state of the target data base in real time.

This may appear to contradict the entire approach to this new design philosophy, since we are now introducing another data base with all the disadvantages of the traditional design, including the requirement to implement shared modify processing (that dreaded mode 1 !!!).

However, upon closer examination, this is not really the case.

In the first place, the actual data base I/O associated with this transaction-in-progress maintenance requirement, is very light (one master data set, no chains, small-sized data entries).

Also, much can be said about utilizing this technique even with the traditional design approach, to simplify data base recoveries and problem resolution, with minimal overhead. This additional data base should probably also contain items such as tables, error messages, and the like - so it can serve a larger, and more general purpose. An appropriate name for this data base could be "SYSDB".

Finally, there exists a school of thought (which the author subscribes to), that discourages placing all your eggs (data) in the same basket (data base). Even with the introduction of TurboIMAGE, and its new (though limited) multi-threading capabilities, many good arguments can be presented against defining all data requirements in a single data base for significant applications. Although these arguments are beyond the scope of this paper, let's just state one of them, which is germane to the issue.

By logically dividing our "data base" into multiple physical IMAGE structures, we gain flexibility in our choice of open modes - which can directly impact overall system throughput. Just because one or two of many data sets require on-line shared modify access, why should we unnecessarily impose the additional overhead requirements upon the entire data base? As long as we keep our data base structures well-defined and logically organized, we can only stand to benefit from the "divide and conquer" approach to IMAGE design.

3. Real-time update requirements.

There may be situations when it is absolutely imperative that updates to certain kinds of data be performed in real-time. For example, consider an order entry application that allocates stock, on-line, as the order taker enters a

customer order during telephone conversation. To ensure that other orders do not deplete non-existent stock, inventories must be updated in real time.

This problem is dealt with in a fashion similar to the "transaction-in-progress" approach, stated above. By simply isolating the data items, requiring real-time updating, in a separate data base, such as SYSDB, we have our solution. A better approach, perhaps (depending upon the application), may be to have a "STOKDB" data base which contains all warehouse inventory-related information. This data base could be opened with mode 1, if new records need to be added during the day, or, better, mode 2 (update only, no data entry adds or deletes), as long as it's understood that new inventory parts could only be added during non-prime hours, while the on-line portion of the application is inactive.

It should be mentioned that, when dealing with mode 2 opens, special care must be taken when defining the data base SCHEMA, to ensure that security is defined at the ITEM level, as well as the data set level. Failure to do so will bring undesired results, since expected write capabilities to data items, may, in fact, be denied to all users except the data base creator. Additional information associated with this phenomenon is left for the reader to investigate.

4. Data Base recovery.

Due to the nature of the IMAGE/message-file design, the window of vulnerability for logical "data base" inconsistency is slightly larger, and, consequently, required data base recovery - necessitated by system or application failures -becomes slightly more complicated- though still quite manageable. We have at our disposal a number of recovery preparation techniques to choose from- which we construct right into the application itself.

Upon system or application failure, all we may be required to do is correct the initial problem and restart the application, which simply continues to process the message file. However, some versions of MPE (particularly, MPE V/P delta 1 and earlier) leave open message files in a corrupted state immediately following a system failure. So, we must employ additional fail-safe mechanisms to protect the integrity of our data.

We may utilize the "transaction-in-progress" approach, mentioned earlier, to our benefit, to enable us to determine exactly which transactions have been **COMMITTED** (output to the message file, and key information placed in SYSDB, such as order number, for example) - but not yet updated in the

target data base.

Once identified, we could (either manually , with QUERY, or automatically, with a user-written program) simply delete logically incomplete transaction(s) from our data base, rebuild the message file, and ask the affected users to reinput the transaction(s), as the application is restarted. (We'll see later how message-file transaction logging can be implemented to offer even better recovery solutions).

We still have to deal with those critical inventory items which have been updated in real time (see point 3, above), but whose corresponding transactions have disappeared into never-never-land because of message file corruption.

An elegant and practical solution to this problem is available to us. Let's use as an example, the order entry application mentioned earlier. In the "STOKDB" data base, we maintain, in the same data entry containing the data items stock-on-hand and qty-allocated, a data item called pending-qty-allocated. The on-line program increments this field (in real-time) at the same time that qty-allocated is incremented. The batch processor program decrements the pending-qty-allocated field while processing the associated transaction from the message file. So, while this field is non-zero, the data base is in a logically inconsistent state.

Another step in our recovery process requires us to run a program which serially processes this data set, and zeros out the pending-qty-allocated field after decrementing the qty-allocated field by its value.

This program can also serve as useful problem-detection tool, and can be affectionately referred to (if so inclined) as the "bug-catcher".

5. Destructive read considerations.

There may be a brief period of time, beginning after the batch processor reads a transaction from the message file, until just prior to it placing the transaction in the data base, during which there is additional risk to data integrity, because of the destructive (by default) nature of message file reads.

For additional protection in those situations (which is probably really only necessary for those installations which experience frequent system or application failures) it's possible to use "FCONTROL" (controlcode 47) to alternately toggle between non-destructive and destructive reads, to

remove the possibility of a failure resulting an incomplete data base update, with no corresponding source transaction in the message file. Of course, if your version of MPE is burdened with the message-file-corruption bug, this technique provides no immediate benefit.

6. Transaction logging.

As many HP3000 users know, transaction logging capability is built in to the IMAGE data base management sub-system.

However, it would be useful, in certain situations, to be able to implement transaction logging for message files.

To do so, one could utilize the same logging mechanism employed by IMAGE - namely, the MPE user logging facility- or another attractive method provided by another subset of the Files System's IPC facility - namely, circular files.

The implementation of circular file technology is very simple and straight-forward, and can offer an excellent message file transaction logging capability.

Circular files are built with the ;CIR option of the :BUILD command. The essence of a circular file is that you can never exhaust its file limit, since records added to a full circular file are wrapped around to the beginning of the file, in a manner transparent to the application (subject to the additional comment below). Thus, a sequential read of the file will produce a chronologically ordered number of records corresponding to the file's capacity. So, to maintain, say, the last 5000 records output to the circular file, one only needs to build it with a capacity of 5000. All associated internal file maintenance is taken care of by MPE.

Now the comment. You should be aware of one very annoying bug associated with circular files, last encountered on MPE V/P delta-1. (the author has not attempted to re-produce the situation on subsequent versions of MPE). If you build a VARIABLE-LENGTH circular file, any attempt to write beyond the end-of-file limit will leave you with a corrupted circular file that can no longer be accessed. The work-around for that is to use fixed-length records, or ensure you never reach end-of-file - by clearing the file at regular intervals, and specifying a large enough capacity to hold the maximum number of records to be accumulated during any of those intervals.

7. Transparent data base backup.

One very attractive benefit inherent with this design approach (at no extra charge), is the ability to perform data base backups while users are still accessing the data base on-line (Transparent Backup).

To accomplish this, one only need terminate the batch processor. Once the data base is accessed by readers only, the data base can be backed up.

However, for those of you using TurboIMAGE (U-mit and beyond), there is a yet another bug (those darn pests !!!) which will, for the time being, deny you the opportunity of doing such backups.

It seems that if a TurboIMAGE data base has ever been opened in a mode other than 5, 6, 7, or 8, since the initial open, that data base is considered to be currently open in a write-type mode - even though it is not - until all accessors of the data base exit from the application. From that point on, on-line users can re-enter their application, and data base backup can begin - as long as the batch processor is not initiated.

It should be noted that, until backup is completed, transactions in progress will accumulate into the message file, until the batch processor is re-introduced. This would mean, for example, that orders placed after batch process interuption, are temporarily unavailable for inquiry or modification.

You may wish to include a facility in your application which is responsible for flagging a terminated batch process condition, in the batch program itself (by placing an appropriate entry in SYSDB), and informing on-line users of this condition at convenient times (say, upon screen form display switching).

NOTE: As of May 1987, this bug is fixed for U-mit (and U-mit deltas) with the T015 patch. Ask your S.E. or the response centre if you really need it. As of this printing, the T015 patch has not been incorporated into MPE version UB delta-2.

8. Batch processor self-tuning.

It's possible to enable the batch processor program to dynamically adjust its own processing priority - based upon external commands (placed into the message file by, say, the application administrator, via a very simple FCOPY job stream similar to the batch process terminator job stream), or upon self-adjusting procedures which change its own

priority according to dynamic conditions.

The latter method is easily implemented, as follows. The on-line program time-stamps all transactions output to the message file. The batch processor, upon reading each transaction from the message file, compares the current time against the time-stamp associated with the transaction. If the time difference is greater than a certain period of time (call it the upper threshold, of, say, 5 minutes), the program could issue a call to the "GETPRIORITY" INTRINSIC to increase its own priority, until such time as the "time-lag" is reduced to a pre-defined lower threshold (say, 3 minutes) at which time another call to "GETPRIORITY" could be issued to revert to the original priority. The batch program needs to be prepped with "PH" capability, in order to accomplish this.

Who needs OPT or APS? We've just built our own poor man's application monitor. On top of that, we now have our own "perpetual motion machine".

9. Problem resolution.

Many of us know how difficult it is to debug on-line programs - not to mention the inconvenience of requiring 2 terminals, in many cases, so that terminal I/O is directed to one terminal (in formatted mode), while the other terminal is used to examine $STDLIST output.

By off-loading a great deal of the processing complexity to a batch process, we now have a much simpler method of identifying and resolving any application problems. We can easily incorporate DISPLAY's, traces, and checkpoints into our batch program (transparent to any on-line user), and examine it's $STDLIST output in hard-copy form, rather than rely on a very keen eye to be able to read cryptic error messages as they roll off a terminal screen.

Not only that, but in many cases we can investigate and solve a problem without requiring on-line users to abandon their activities while diagnosis is performed, and remedial action is taken.

This is almost like being able to change a flat tire while still screeching down the highway to make it to the bank before it closes.

10. Other considerations.

You should be aware that the described method of data base

access (i.e. mode 6 open) is not yet common in the HP3000 community.

When you begin to implement applications employing these techniques, you may discover - all of a sudden - that some of your favourite software tools and general-purpose applications don't work as expected any more.

For example, DBLOADNG, a popular INTEREX contributed library IMAGE data base analysis tool, will not run concurrently with mode 6 users of the data base, while it was perfectly content to co-exist with your former application design. The reason for this is that DBLOADNG opens the specified data base with mode 5, which is incompatable with mode 6 opens (but compatable with mode 1 and/or mode 5 opens). Likewise, with HPACCESS, a data base extraction tool, from Hewlett-Packard. These are just two of many examples.

On the other hand, HOWMESSY, a data base analysis tool similar to DBLOADNG, from Robelle Consulting, very cleverly attempts to open the specified data base with mode 6 after an unsuccessful mode 5 attempt.

If you use other software products which are unable to co-exist with your newly-designed applications due to this easily-correctable problem, you may wish to consider asking your software vendor to offer a solution in the next release.

SUMMARY

The careful and intelligent use of a very powerful and flexible subset of MPE's file system, the Interprocess Communication Facility (IPC), provides significant new opportunities for raising IMAGE-based application design technologies to more advanced levels.

Substantial improvements in programming productivity, execution performance, system availability, and problem identification and resolution are at our immediate disposal.

Effective HP3000 System Management
or
Don't Clutter Your Desk

by

Gilles Schipper
G. Schipper & Associates Inc.
P.O. Box 310
Thornhill, Ontario L3T 4A2
Canada

ABSTRACT

I hope your HP3000 is not like my desk at home - everything piled on top, where finding what you want is time-consuming, and even the most sensitive material is within easy reach of anybody that dares search the tabletop.

Proper and disciplined Account Management offers significant benefits. These include:

(1) Resource Optimization (Human and Machine)
(2) System Performance Optimization
(3) Quicker Program Development
(4) Faster Problem Resolution
(5) Increased Security
(6) Potential for Unattended Backup
(7) ... and much more

This paper presents a methodology for one very important aspect of effective system management - account management- together with examples for implementing numerous resource optimization techniques - made possible with (and only with) proper account management.

INTRODUCTION

As system manager, it's likely that your primary objective is to maximize system throughput and availability, with a minimum expenditure of human and machine resources. In other words, increasing productivity.

Consider your HP3000 computer system, including the system domain discs, as a large reservoir of information storage capacity. Think of the HP3000 as comprising one or more entities known as accounts. For the time being, let's ignore

the other major components of the HP3000 - such as CPU, main memory, and non-disc peripheral devices.

A sound (and largely, common-sense) methodology for defining, creating, maintaining, and managing these accounts is the first step to successful system management.

ACCOUNT MANAGEMENT

Surely, there must be a law somewhere (perhaps some corollary of Murphy's Law) which states that the fewer entities needed to be managed and controlled, the easier is the task of management and control.

This assumes, of course, that it is desireable to control a given set of entities - and this is certainly the case for accounts on the HP3000. An intimate knowledge of the existence of all HP3000 accounts is necessary for proper disc space management, security management, and, generally, HP3000 system management - as we will learn.

By eliminating unnecessary accounts, we gain several benefits:

(1) Since each account is a "window" into the system, available to anybody with terminal access to the computer, including modem access, reducing the number of windows increases security.

(2) Since each account consumes directory entries, reducing the number of them - as well as corresponding users, groups, and files, which also occupy directory space - has a positive effect upon conserving a limited resource. Some may argue that with a directory maximum of 60,000 sectors, compared to the previous MPE III maximum of 6,000 sectors, this no longer represents a significant problem. However, with larger and larger systems and disc drives continually being developed, 60,000 sectors may soon again become a limiting factor - particularly to those who choose to ignore this exhaustible resource.

(3) Accounts not only occupy directory space, they presumably also contain files which, in turn, occupy disc space. Redundant files not only waste disc space, they also tend to be backed up to tape on a regular basis (say, once a week - I'll have more to say about backups later). What can be more fruitless than having an operator mounting an extra tape or two EVERY WEEK to back up files which will never be used?

(4) Finally, getting back to the major point, fewer accounts

to manage simplify and streamline the job of system management.

Assuming we wish to prune our system of unused accounts, how do we recognize which accounts are bloating our discs and directory?

First, examine the inventory of accounts on your HP3000. A simple :REPORT X.@ command (requiring SM capability) will display all accounts on your system (as well as all groups named X).

Possibly, many accounts are there only as the result of HP's software installation procedures, when upgrading to a new version of the MPE.

Following are some accounts which may have been created only for the purpose of operating system upgrades, that may or may not represent clutter on your system.

SUPPORT

This account used to be the account used by HP field personnel (Customer Engineers and System Engineers) to perform diagnostics, preventative maintenance, and general troubleshooting. It is also used during the software installation process and software update process to create the diagnostic utility subsystem (DUS) tape, which should be kept in a very safe and accessible location. You should hope you will never need to use the DUS tape - and you should also ensure you have one in the rare event it is needed. However, once the operating system has been installed and/or updated, the SUPPORT account need not remain on your system and should be purged. The TELESUP account is now being used by HP field personnel for support purposes. By purging the SUPPORT account, you will not only save between 30,000 and 56,000 sectors of disc space (depending on MPE version), you will also free up a large number of directory entries formerly occupied by the large number of groups and files in the SUPPORT account. If, for any reason whatsoever, they need to be restored, they can be retrieved from the latest FOS tape.

ITF3000

This account is used for the optional interactive training facility package, used for the purposes of on-line, interactive training of operators and users of the HP3000. If you don't know whether or not you have purchased this product (as system manager, you should know), simply check if the ITF3000 account contains any files. If not, you don't

have the optional product, the ITF3000 account is serving no useful purpose, and it should be purged.

HPPL85, HPPL87, HPPL89, HPPL96, HPOFFICE, HPWORD

These accounts are used by a variety of office automation products, such as HPWORD, TDP, HPSLATE, HPDESK, PRINT CENTRAL, HPDRAW, EZCHART, DSG, etc.

If you do not subscribe to any of these products, purge these accounts. If you do have one or more of these products, if the account in question has no files, purge the account.

It should be noted that HPPL87 holds various source code files for the program ENTRY.PUB.SYS. If you wish to use these source programs to assist in the development of VPLUS programming, you may wish to keep this account, or move the files wanted to another account, then purge HPPL87. Even if you purge the account, you can always retrieve any necessary files later from your FOS tape.

The HPWORD account is provided as a log-on account for HPWORD users, and is not used to house any specific HP files. If you do not wish to isolate your HPWORD users in a single account, or if you wish to use a different account for that purpose, then purge the HPWORD account.

RJE

As the name implies, this account presumably is used by the RJE/3000 subsystem. Later, we will learn how to recognize whether, in fact, it is required - and if not, should be purged.

TELESUP

This account is the current replacement for the old SUPPORT account mentioned above. In most cases, this account can serve an extremely useful purpose for Hewlett-Packard support personnel, and you, the HP customer.

Apart from some very useful UNSUPPORTED utilities included in that account (such as BULDACCT, TUNER, FLUTIL, and DIRPUR - to name only a few), it also contains data and programs utilized by HP's predictive support and HPTREND services.

If you do not have a 7976 tape drive, by all means purge the more than 80 files with the name RT##### in the HP32340 group. While these files may not occupy a tremendous amount of disc space (900 sectors), they consume many directory

entries and will slow down your backups.

A final note about this account: The default password of the TELESUP account is HPONLY. The MGR user password is MGR, and most of the groups within the account have group passwords corresponding to the group name. Also, this account possesses PM capability, meaning that anybody who is able to log on to this account virtually has unlimited access to your entire system. This is no large secret in the HP community - so, one of your first tasks should be to change the password at the account level of this account. You will also need to appropriately modify the predictive support and HPTREND job streams to reflect the new password. More on security is presented later in this paper.

SYS

This account, of course, contains all necessary software to allow MPE, and all HP subsystems and utilities to operate. You could not bring up the system without this account.

However, if you are not a user of a laser printer, you will probably not have much use for the more than 17,000 sectors of disc space included in the group HPENV.SYS.

Other groups in SYS not required on a long-term basis, include the CREATOR group. Apart from MPE installations and upgrades, files in this group are needed only if necessary to revert back to a prior operating system version.

Many other groups included in the SYS account may also be redundant. We will learn how to recognize them shortly.

If you're really short on disc space, you may wish to prune even the PUB.SYS group of unnecessary files. Space does not allow complete identification of unneeded PUB.SYS files here, but some clues to a method for their identification are presented later.

You should be aware that the process of removing redundant accounts, users, groups, and files must be repeated after each MPE operating system upgrade. Of course, it's a snap after the first time.

OTHER ACCOUNTS

What about all of the other accounts occupying our valuable disc space and directory, requiring tape storage, and otherwise requiring the attention of the system manager. How do we know whether these accounts, and the files, groups, and users contained within, actually need to kept on-line.

Before suggesting methods for determining the relevance of these accounts with respect our operating environment, let's introduce the important concept of **ACCOUNT TYPE**, which will be referred to several times over in the rest of this article.

Think of each account on your system as belonging to one of two of the following account types:

(1) **READ-ONLY** Account

A read-only account is one which is created on your system solely for the purpose of housing files which are read (or executed) by users of the system. Files located in read-only accounts are generally not modified, and read-only accounts are generally not logged on to on a regular basis.

Examples of read-only accounts include BRADMARK, BWRUG, CAROLIAN, COGNOS, DISC, INFOSYS, INTX2, MAINLIB, NOMLIB, NSD, PSS, REGO, ROBELLE, SECURITY, TELESUP, TYM, and VESOFT.

As you may have guessed, read-only accounts typically contain third-party or contributed-library software, and files contained within them do not change on a regular basis. Modifications of files in read-only accounts almost always correspond to the receipt of software update tapes from third-party vendors (usually accompanied with an invoice for maintenance and support).

The above examples are the most obvious ones in the read-only account category. A not so obvious example is the SYS account. Strictly speaking, this account does not normally qualify as a read-only account - for two reasons.

First, files within the SYS account do change and are created regularly. However, the files in question are limited, and well-defined. In most cases, these include COMMAND.PUB.SYS, and LOG####.PUB.SYS files.

Second, the SYS account is typically regularly logged on to - most often by OPERATOR.SYS and MANAGER.SYS. However, there is no good reason for that to be true on your system. In fact, particularly beginning with MPE V/E, T-mit, it is quite possible - indeed desireable - to effectively transform the SYS account into a read-only account. By so doing, you minimize the possibility of creating non-HP files in the SYS account, while increasing control of your system. Another benefit is associated with system backup strategies - with more details to follow.

By creating the SYSTEM account, and including the users MGR and OPER, all system management and operations activities

can be undertaken in an account completely independent of the SYS account. More details of what the SYSTEM account should look like are provided later.

(2) **LOG-ON** Account

Accounts which are regularly logged on to, are classified as log-on accounts. These accounts are simply those that are NOT read-only accounts, according to the definition above.

Examples may include, PROD, PAYROLL, SYSTEM, DEVL, TEST, etc.

Sometimes, it may not be entirely clear whether an account is read-only or log-on. Apart from the SYS account described above, you may have purchased an application package from, say, MCBA, which includes source program, executable code, and data base files in the MCBA account. In such cases, you should examine those accounts, and at least try to identify read-only groups within them. Better still, if possible, attempt to separate your company-specific data files from the generic application-specific files into different accounts so that your files are clearly distinguishable from the software vendor's files. This increases the control you have upon your accounts, and minimizes the potentially damaging impact of future software updates upon your system. Incidentally, the latest version MCBA modules (version 3) now recognizes this advantage, and allows you to more easily separate the software from your data.

Furthermore, where an account is clearly offered as a read-only account, keep it that way. For example, do not create new users or groups in the COGNOS or VESOFT accounts and allow users to log on to these accounts and create their own files within them. This only leads to reduced control and security on your system.

Now, back to the original question. Which accounts should rightly occupy permanent residence in our discs, and which should be evicted (presumably, after first storing them to tape - just in case).

To answer that question, we need to know whether a given account is a legitimate log-on account or read-only account, assuming we don't already know.

To determine whether or not an account is log-on, we can do one of two things. First, change the password at the account level, and wait until somebody hollers. If you don't hear from anybody for some reasonable period of time, and batch jobs do not suddenly abort because of invalid passwords, you may safely assume the account is NOT a log-on one. A second

method is to issue a RESETACCT command to clear the account cpu and connect-time accumulators and watch them over a period of time to see if they remain at zero. Incidentally, it's probably not a bad idea to do this on a regular basis at a well-defined time interval (say, weekly, as part of the full backup procedure), enabling you to identify and monitor active log-on accounts regularly.

If you've ruled out the account as a log-on type, it only remains to determine whether or not it's a read-only one. To do that, you need a tool such as MPEX (from VESOFT), or lots of paper and lots of time. With MPEX (%LISTF @.@.ACCT,3), or LISTDIR5 (>LISTF @.@.ACCT) simply find out when all files in the account were last accessed. The most recent last-accessed-date represents the last time any file in the account were accessed. Armed with this knowledge, you are now better equipped to determine whether the account is read-only or not. It should be pointed out here that the effectiveness of this method is dramatically reduced if system reloads performed at your site are of the RELOAD ACCOUNTS type, followed by a RESTORE @.@.@ (the preferred method) WITHOUT the OLDDATE option. Omission of OLDDATE from the RESTORE command resets the access-date in the file labels to the date of the restore.

So, if an account does not meet the criteria of log-on or read-only, according to the definitions above, it can safely be purged. (Remember, of course, that you have first stored them to tape - twice, or once plus a VALIDATE - just in case).

ACCOUNT STRUCTURES

Now that we're able to separate the wheat from the chaff of our existing accounts, how do go about setting up new accounts.

First, let's return to the issue of transforming the normal log-on account SYS into a read-only account.

To do this, we must set up an alternate account (say, SYSTEM) from which to perform normal system management and operations activities - as well as preventing SYS from being logged on to (except for rare occasions, such as performing operating system updates).

For those HP3000 users on the T-mit version of MPE or later, the standard log-on for OPERATOR.SYS following any system startup (WARMSTART, COOLSTART, COLDLOAD, UPDATE, or RELOAD), can be overridden simply by creating the file SYSSTART.PUB.SYS (one of those rare occasions when you need

to log on to the SYS account), with any text editor - to include the following:

```
STARTUP
JOBFENCE 14
STREAMS 10
STARTSESS 20;OPER/OPERPASS.SYSTEM;HIPRI;NOWAIT[;PRI=BS]
****
```

These commands will be executed immediately after any system startup - automatically, by MPE (except after a RELOAD with the NULL or ACCOUNTS option, since SYSSTART won't be on your system). The STARTSESS command will also prevent MPE from logging on the system console as OPERATOR.SYS, thus accomplishing the desired result. An alternate method of achieving this, for those with versions of MPE earlier than T-mit, would be to purge the user OPERATOR.SYS.

Incidentally, it behooves you to minimize the functions performed at system startup time, so that control of your system is not taken away from you or the system supervisor. A "START" UDC for OPER.SYSTEM can be used to set up the normal operating environment - when, and only when, you wish to do so.

Also, don't forget to alter the file level security of SYSSTART.PUB.SYS so that users cannot gain read access to that file, since it contains the password for OPER.SYSTEM. Do this as follows:

```
:ALTSEC SYSSTART;(R,L,X,W,A:CR)
```

The SYSTEM Account

What should the SYSTEM account look like? Since it must, for many purposes, replace the SYS account, it must have the same capabilities. So, we create the SYSTEM account as follows:

```
:NEWACCT SYSTEM,MGR;ACCESS=(R,L,X,W,A:ANY);MAXPRI=BS; &
: CAP=SM,AM,AL,GL,DI,OP,PH,DS,MR,PM,IA,BA,CS,ND,SF,UV,CV, &
:      LG,PS,NM,NA
```

The latter three capabilities, PS,NM, and NA are new, beginning with T-mit.

We choose MGR as the account manager user for this account, and OPER as the system supervisor.

The entire profile of a "suggested" SYSTEM account is illustrated with the following job stream:

```
!JOB SYSTEMJ,MGR.SYSTEM;HIPRI
!COMMENT ****************************************************
!COMMENT *                    ** SYSTEMJ  **
!COMMENT *
!COMMENT *     THIS JOB CREATES THE SYSTEM ACCOUNT STRUCTURE.
!COMMENT * FOR THIS JOB TO WORK, THE 'SYSTEM' ACCOUNT MUST
!COMMENT * FIRST BE CREATED WITH ALL CAPABILITIES, AS
!COMMENT * FOLLOWS:  NEWACCT SYSTEM,MGR;CAP=SM,AM,..etc.
!COMMENT ****************************************************
!TELLOP **SYSTEMJ** JOB STARTED
!ALTACCT SYSTEM  ;ACCESS=(R,L,X,W,A:ANY);MAXPRI=BS
!COMMENT ****************************************************
!COMMENT *             Create USER structure
!COMMENT ****************************************************
!COMMENT * THE NEXT ALTUSER COMMAND WORK FOR ALL MPE VERS.
!COMMENT ****************************************************
!ALTUSER MGR;CAP=SM,AM,AL,GL,DI,OP,PH,DS,MR,PM,IA,BA,CS, &
!               ND,SF,UV,CV,LG
!COMMENT ****************************************************
!COMMENT *  THE NEXT ALTUSER COMMAND WILL ONLY SUCCEED FOR
!COMMENT *  MPE VERSION T-MIT AND LATER.
!COMMENT ****************************************************
!
!CONTINUE
!ALTUSER MGR;CAP=SM,AM,AL,GL,DI,OP,PH,DS,MR,PM,IA,BA,CS, &
!               ND,SF,UV,CV,LG,PS,NM,NA
!ALTUSER MGR; HOME=MGR
!ALTUSER MGR; MAXPRI=BS
!ALTUSER MGR; PASS=SUPERMAN
!
!CONTINUE
!NEWUSER OPER
!ALTUSER OPER;CAP=IA,BA,ND,SF,CS,LG,CV,OP,GL
!ALTUSER OPER;HOME=OPER
!ALTUSER OPER;MAXPRI=BS
!ALTUSER OPER;PASS=BIGSHOT
!
!COMMENT ****************************************************
!COMMENT *              Create GROUP structure
!COMMENT ****************************************************
!ALTGROUP PUB;CAP=IA,BA,PH,MR,DS,PM
!ALTGROUP PUB;ACCESS=(R,X,L:ANY;W,A,S:AL)
!COMMENT ****************************************************
!COMMENT * THE CONTINUE COMMAND PRECEDING THE NEWGROUP
!COMMENT * COMMANDS, WHICH, IN TURN ARE FOLLOWED BY ALTGROUP
!COMMENT * COMMANDS ENABLE THIS JOB STREAM TO BE RE-USABLE.
!COMMENT ****************************************************
!
!comment  dat contains system data files (sleepcom, etc.)
!CONTINUE
!NEWGROUP DAT
```

```
!ALTGROUP DAT;CAP=IA,BA
!ALTGROUP DAT;ACCESS=(R,X,L:ANY;W,A,S:AL)
!
!comment  db contains image data bases used system-wide
!CONTINUE
!NEWGROUP DB
!ALTGROUP DB ;CAP=IA,BA
!ALTGROUP DB ;ACCESS=(R,X,L,W,A:ANY;S:AL)
!
!comment  dbl contains LOGGED image data bases
!CONTINUE
!NEWGROUP DBL
!ALTGROUP DBL;CAP=IA,BA
!ALTGROUP DBL;ACCESS=(R,X,L,W,A:ANY;S:AL)
!
!comment  doc contains system documentation files
!CONTINUE
!NEWGROUP DOC
!ALTGROUP DOC;CAP=IA,BA
!ALTGROUP DOC;ACCESS=(R,X,L:ANY;W,A,S:AL)
!
!comment  frm contains VPLUS fast form files for system use
!CONTINUE
!NEWGROUP FRM
!ALTGROUP FRM;CAP=IA,BA
!ALTGROUP FRM;ACCESS=(R,L,X:ANY;W,A,S:AL)
!
!comment  job contains system job streams (fullbuj, etc.)
!CONTINUE
!NEWGROUP JOB
!ALTGROUP JOB;CAP=IA,BA
!ALTGROUP JOB;ACCESS=(X,L:AC;R,W,A,S:AL)
!
!comment  lib contains skeletons, copylibs, etc.
!CONTINUE
!NEWGROUP LIB
!ALTGROUP LIB;CAP=IA,BA
!ALTGROUP LIB;ACCESS=(R,X,L:ANY;W,A,S:AL)
!
!comment  prv contains contributed or in-house programs
!comment  which use PM capability
!CONTINUE
!NEWGROUP PRV
!ALTGROUP PRV;CAP=IA,BA,PH,MR,DS,PM
!ALTGROUP PRV;ACCESS=(X,L:ANY;R,W,A,S:AL)
!
!comment  src contains source for contributed or in-house
!comment  programs (found in PRV or UTL groups). It also
!comment  includes any VPLUS form files or image schemas
!comment  for data bases or fast form files used system-
!comment  wide.
!CONTINUE
```

```
!NEWGROUP SRC
!ALTGROUP SRC;CAP=IA,BA
!ALTGROUP SRC;ACCESS=(R,X,L,W,A,S:AL)
!
!comment  udc contains system-wide udc files, and udc files
!comment  used by users in the SYSTEM account. in a very
!comment  strictly-controlled environment, it could house
!comment  all udc files used throughout the HP3000.
!CONTINUE
!NEWGROUP UDC
!ALTGROUP UDC;CAP=IA,BA
!ALTGROUP UDC;ACCESS=(R,X,L:ANY;W,A,S:AL)
!
!comment  usl contains usl's contained in programs in
!comment  PRV or UTL. also usl's used by SL.PUB, SL.PRV,
!comment  or SL.UTL. also, any RL's are in this group.
!CONTINUE
!NEWGROUP USL
!ALTGROUP USL;CAP=IA,BA
!ALTGROUP USL;ACCESS=(R,L:ANY;X,W,A,S:AL)
!
!comment  utl contains contributed or in-house utility
!comment  programs.
!CONTINUE
!NEWGROUP UTL
!ALTGROUP UTL;CAP=IA,BA,PH,MR,DS
!ALTGROUP UTL;ACCESS=(X,L:ANY;R,W,A,S:AL)
!
!comment  vfc contains system-wide vfc files
!CONTINUE
!NEWGROUP VFC
!ALTGROUP VFC ;CAP=IA,BA
!ALTGROUP VFC ;ACCESS=(R,L,X:ANY;W,A,S:AC)
!
!comment  wlc contains system welcome message files
!CONTINUE
!NEWGROUP WLC
!ALTGROUP WLC ;CAP=IA,BA
!ALTGROUP WLC ;ACCESS=(R,L,X:ANY:W,A,S:AC)
!
!comment  mgr contains the SM's work-in-progress files
!CONTINUE
!NEWGROUP MGR
!ALTGROUP MGR ;CAP=IA,BA,PH,MR,DS,PM
!ALTGROUP MGR ;ACCESS=(R,X,L,W,A,S:AL)
!
!comment  oper contains oper's work-in-progress files
!CONTINUE
!NEWGROUP OPER
!ALTGROUP OPER;CAP=IA,BA
!ALTGROUP OPER;ACCESS=(R,X,L,W,A,S:GL)
!TELLOP **SYSTEMJ** JOB COMPLETED SUCCESSFULLY
```

```
!EOJ
```

Notice that the 2 users of the SYSTEM account - namely, MGR and OPER - each have their namesake as home groups. This standard is maintained throughout when building accounts over which we exercise complete control.

If we use an office desk as our analogy to our HP3000 account, we discourage untidy desks by assigning to each account user a drawer (group) corresponding to the user's name. It then becomes a routine matter to encourage minimization of file space in each user's work-in-progress drawer (group) - and to easily identify those that violate this objective.

Most office desks - and most HP3000 accounts - typically accumulate papers (files) on the desktop (PUB group).

Notice also that each group of documents (files) is housed in appropriate file drawers (groups) and each drawer (group) is equipped with appropriate security (group access attributes).

In this example of the SYSTEM account, as in examples which follow, there is NEVER a requirement to assign passwords at the group level - which would only become another password which never changes, and thus becomes less effective. This is possible because, at the group level, we only allow, for example, write access to a user with AL capability, enabling, in this case, only the system manager to write (or purge) files in the account (other than the OPER, VFC, and WLC groups). Similarly, the JOB group denies read access to any other than the SM, but enables eXecute access to OPER.

The principle followed in setting up the SYSTEM account is to have each group appropriately named and secured, and with only the minimum capabilities necessary assigned.

Since account and group access attributes are carefully and properly devised, there is rarely a need to compromise system security by regularly RELEASEing files. This would only happen in extremely rare circumstances.

Assigning the system supervisor OPER.SYSTEM maxpri=BS capability is recommended as long the user of this capability is fully aware of its possible abuse, and can be trusted with this powerful tool. It can be very convenient to have the console operator sign on with MAXPRI=BS in the event of apparent system hangs.

Since the SYSTEM account is considered to be a log-on account, a password is placed at the user level only - and

not the account level. More on security later.

All system management and operations activities can be carried on from the SYSTEM account, leaving the SYS account as a read-only account.

OTHER ACCOUNTS

Now, to set up other accounts - say for system development and production, we generally follow the same guidelines as the SYSTEM account structure, with some very important exceptions.

First, since we are not attempting to duplicate the system management and operations capabilities of the SYS account, we do not need (nor should we include) many capabilities, such as SM, PM, DI, to mention a few.

Second, our development and production accounts do not have the PRV and OPER groups.

An example of a TEST account structure creation job stream follows:

```
!JOB ACCTCRJ,MGR.TEST
!COMMENT ************************************************
!COMMENT *                    ** ACCTCRJ **
!COMMENT *
!COMMENT *    THIS JOB CREATES THE STANDARD LOG-ON ACCOUNT.
!COMMENT ************************************************
!TELLOP **ACCTCRJ** CREATE LOG-ON ACCOUNT STRUCTURE
!ALTUSER MGR;HOME=MGR
!ALTUSER MGR;PASS=MGRPASS
!ALTUSER MGR;CAP=IA,BA,PH,MR,SF,ND,AM,AL,GL,LG,CS,OP
!CONTINUE
!NEWUSER USER
!ALTUSER USER;HOME=USER
!ALTUSER USER;PASS=USERPASS
!ALTUSER USER;CAP=IA,BA,SF,ND,LG,CS,GL
!ALTGROUP PUB      ;ACCESS=(R,L,X:ANY;W,A,S:GL)
!ALTGROUP PUB      ;CAP=IA,BA
!CONTINUE
!NEWGROUP LIB
!ALTGROUP LIB      ;ACCESS=(R,L,X:ANY;W,A,S:GL)
!ALTGROUP LIB      ;CAP=IA,BA
!CONTINUE
!NEWGROUP DAT
!ALTGROUP DAT      ;ACCESS=(R,L,X,W,A:ANY;S:GL)
!ALTGROUP DAT      ;CAP=IA,BA
!CONTINUE
```

```
!NEWGROUP DB
!ALTGROUP DB       ;ACCESS=(R,L,X,W,A:ANY;S:GL)
!ALTGROUP DB       ;CAP=IA,BA
!CONTINUE
!NEWGROUP DBL
!ALTGROUP DBL      ;ACCESS=(R,L,X,W,A:ANY;S:GL)
!ALTGROUP DBL      ;CAP=IA,BA
!CONTINUE
!NEWGROUP DOC
!ALTGROUP DOC      ;ACCESS=(R,L,X:ANY;W,A,S:GL)
!ALTGROUP DOC      ;CAP=IA,BA
!CONTINUE
!NEWGROUP FRM
!ALTGROUP FRM      ;ACCESS=(R,L:ANY;X,W,A,S:GL)
!ALTGROUP FRM      ;CAP=IA,BA
!CONTINUE
!NEWGROUP FGL
!ALTGROUP FGL      ;ACCESS=(R,L,X:ANY;W,A,S:GL)
!ALTGROUP FGL      ;CAP=IA,BA
!CONTINUE
!NEWGROUP JOB
!ALTGROUP JOB      ;ACCESS=(L,X:ANY;R,W,A,S:GL)
!ALTGROUP JOB      ;CAP=IA,BA
!CONTINUE
!NEWGROUP MGR
!ALTGROUP MGR      ;ACCESS=(R,L,X,W,A,S:GL)
!ALTGROUP MGR      ;CAP=IA,BA
!CONTINUE
!NEWGROUP PRG
!ALTGROUP PRG      ;ACCESS=(X:ANY;R,L,W,A,S:GL)
!ALTGROUP PRG      ;CAP=IA,BA,PH,MR
!CONTINUE
!NEWGROUP SRC
!ALTGROUP SRC      ;ACCESS=(R,L,X,W,A,S:GL)
!ALTGROUP SRC      ;CAP=IA,BA
!CONTINUE
!NEWGROUP USL
!ALTGROUP USL      ;ACCESS=(R,L,X:ANY;W,A,S:GL)
!ALTGROUP USL      ;CAP=IA,BA
!CONTINUE
!NEWGROUP UDC
!ALTGROUP UDC      ;ACCESS=(R,L,X:AC;W,A,S:GL)
!ALTGROUP UDC      ;CAP=IA,BA
!CONTINUE
!NEWGROUP UTL
!ALTGROUP UTL      ;ACCESS=(R,L,X:ANY;W,A,S:GL)
!ALTGROUP UTL      ;CAP=IA,BA
!CONTINUE
!NEWGROUP USER
!ALTGROUP USER     ;ACCESS=(R,L,X,W,A,S:GL)
!ALTGROUP USER     ;CAP=IA,BA
!TELLOP **ACCTCRJ** JOB COMPLETED SUCCESSFULLY
```

```
!EOJ
```

Notice again, the two users, MGR and USER each have home group names equal to the user name.

The GL capability is used here to enable full access for USER to only USER's home group, without the need for group passwords. For this to work, USER must be given GL capability, which is enabled only while USER is logged in to USER's home group.

Only the account manager, MGR, can read source programs (SRC) or executable programs (PRG), since allowing read capability, even on program files (which could contain data base passwords, for example) could represent a breach of security.

Similarly, files in the JOB group can only be read by the AM (or SM), but STREAMed by anybody, including, of course, OPER.SYSTEM. It becomes a simple matter to prevent non-account users from streaming job streams in the TEST account by simply modifying the group level access.

SECURITY

The security provisions which form part the MPE operating system come in two basic flavors (other flavors exist, but are not mentioned in detail here). The first is responsible for preventing unauthorized terminal access to the HP3000. This security provision, known as log-on security, is implemented in the form of passwords at the account, user, and group levels. The second, referred to as file system security, is responsible for preventing a given legitimate HP3000 user (legitimate, because of successful log-on) from accessing another user's files.

File system security is effectively addressed by devising well-reasoned access attributes at the account and group levels, as shown in the sample job streams above.

Many HP3000 security schemes break down due to improper maintenance of log-on security. This is due mostly to poor account structure organization and the general difficulty associated with changing passwords. Passwords are rarely changed because of resulting side-effects associated with batch job streams, which often contain embedded passwords-and, accordingly, must be modified coincidentally with the changing of account, user, and group passwords. The end result - passwords rarely change, and log-on security loses its effectiveness over time as more and more people become familiar with the existing passwords.

One solution to this problem is a third-party software package from VESOFT called SECURITY/3000, which effectively addresses not only the log-on security problem, but also a host of other security issues ranging from remote terminal access to enforced password maintenance.

However, even with such a well-designed and complete tool as SECURITY/3000 (and even more so without), it pays to be organized.

To illustrate the point, let's assume our system contains the following accounts, in alphabetical order:

```
BRADMARK    (read-only)
COGNOS      (read-only)
DISC        (read-only)
FINANCE     (log-on)
HPOFFICE    (read-only)
INFOSYS     (read-only)
PERSONEL    (log-on)
PROD        (log-on)
REGO        (read-only)
ROBELLE     (read-only)
SCHIPPER    (read-only)
SECURITY    (read-only)
SYS         (read-only)
SYSTEM      (log-on)
TECH        (read-only)
TELESUP     (read-only)
TEST        (log-on)
TYM         (read-only)
VESOFT      (read-only)
```

Read-only accounts should have passwords at the account level only - for the sole purpose of denying log-on access to them by all except the system manager. Log-on accounts, on the other hand, should NOT have passwords at the account level - but, rather, at the user level only. An account password quickly loses its effect when known by many users in the account, and only serves to frustrate the user by requiring knowledge of two passwords to successfully log on (1 at the account level, and 1 at the user level to distinguish different users). As demonstrated earlier, the requirement for group passwords can be eliminated by judicious use of GL and AL capabilities. The only exception would be where group usage statistics (such as connect time and CPU usage) are utilized, and it becomes important to prevent users from even logging on to other groups.

The read-only accounts and the user MGR.SYSTEM can, and SHOULD, all have the same password. Otherwise, it's like the

security officer of a large building having to carry around a large number of different keys to open each lock, instead of being equipped with a single master key.

To effectively change the system manager password, one only needs a job stream, such as the one following, to do it. This job stream uses MPEX to accomplish part of the job. Without it, or a suitable replacement, you need to manually edit the files in JOB.SYSTEM to change all occurrences of OLDSMPAS to NEWSMPAS. Of course, we are assuming that all jobs logging on to the SYSTEM account are located in JOB.SYSTEM - the result of sound account management techniques. (With SECURITY/3000, the need to edit job stream files for password maintenance is virtually eliminated, since passwords are, for most intents and purposes, not required in these job streams.)

```
!JOB SMPASJ,MGR.SYSTEM;HIPRI;PRI=CS;OUTCLASS=LP,1
!COMMENT ******************************************************
!COMMENT *                    ** SMPASJ **
!COMMENT *
!COMMENT *     THIS JOB CHANGES THE SYSTEM MANAGER PASSWORD
!COMMENT * WHEREVER IT OCCURS (MGR.SYSTEM, ALL READ-ONLY
!COMMENT * ACCOUNTS, AND ALL FILES IN JOB.SYSTEM).
!COMMENT *     MPEX IS USED TO GLOBALLY CHANGE ALL JOB
!COMMENT * STREAM FILES IN JOB.SYSTEM.
!COMMENT ******************************************************
!TELLOP **SMPASJ** JOB STARTED
!TELL MGR.SYSTEM;SMPASJ STARTED
!TELL MGR.SYSTEM;^G^G^G  ^G^G^G  ^G^G^G  ^G^G^G
!TELL MGR.SYSTEM;STAY LOGGED ON UNTIL JOB COMPLETES,
!TELL MGR.SYSTEM;AND CHECK RESULTS
!TELL MGR.SYSTEM;^G^G^G  ^G^G^G  ^G^G^G
!RUN MPEX.PUB.VESOFT
SET DEFAULT,@,!
EDIT @.JOB,C"/OLDSMPAS","/NEWSMPAS",ALL
ALTFILE GOD.PUB.VESOFT,LOCK=NEWSMPAS
EXIT
!ALTUSER MGR      ;PASS=NEWSMPAS
!CONTINUE
!ALTACCT BRADMARK;PASS=NEWSMPAS
!CONTINUE
!ALTACCT COGNOS   ;PASS=NEWSMPAS
!CONTINUE
!ALTACCT DISC     ;PASS=NEWSMPAS
!CONTINUE
!ALTACCT HPOFFICE;PASS=NEWSMPAS
!CONTINUE
!ALTACCT INFOSYS  ;PASS=NEWSMPAS
!CONTINUE
!ALTACCT REGO     ;PASS=NEWSMPAS
```

```
!CONTINUE
!ALTACCT ROBELLE ;PASS=NEWSMPAS
!CONTINUE
!ALTACCT SCHIPPER;PASS=NEWSMPAS
!CONTINUE
!ALTACCT SECURITY;PASS=NEWSMPAS
!CONTINUE
!ALTACCT SYS     ;PASS=NEWSMPAS
!CONTINUE
!ALTACCT TECH    ;PASS=NEWSMPAS
!CONTINUE
!ALTACCT TELESUP ;PASS=NEWSMPAS
!CONTINUE
!ALTACCT TYM     ;PASS=NEWSMPAS
!CONTINUE
!ALTACCT VESOFT  ;PASS=NEWSMPAS
!TELLOP **SMPASJ** JOB COMPLETED SUCCESSFULLY
!TELL MGR.SYSTEM;SMPASJ COMPLETED
!TELL MGR.SYSTEM;CHECK RESULTS PRIOR TO LOGGING OFF
!TELL MGR.SYSTEM;^G^G^G  ^G^G^G  ^G^G^G
!TELL MGR.SYSTEM;
!SET STDLIST=DELETE
!EOJ
```

Password maintenance for each log-on account is also best handled with appropriate job streams - one job stream to modify the AM user password (MGR) wherever it occurs (i.e. the user MGR, and the files in JOB.ACCT) - and a second job stream to modify the other user passwords.

Remember, job streams are much less prone to errors, automatically provide an audit trail of commands executed, and can be re-used over and over again.

BACKUP STRATEGIES

We've learned how to identify and classify accounts (read-only vs. log-on) and how to organize them for effective security and maintenance management.

We can now realize additional benefits by optimising our backup strategies to take advantage of our meticulous organization.

Many shops employ a backup methodology which calls for a weekly full backup (@.@.@ files, 0 date), and a daily partial which backs up all files changed since the previous full (or, in some cases, the previous partial). Often, this could result in very inefficient use of both human and machine resources.

For those shops that leave unnecessary accounts on the

system, time and tape is wasted to back them up. Also, since read-only accounts rarely change, they are often needlessly backed up, albeit only once a week.

The real potential for savings can be realized for those partial backups, which take place 4, 5, or 6 times weekly.

Since most files which are backed up on a daily basis, and those which generally occupy most of the backup tapes are IMAGE data sets, we can very effectively virtually decimate the time and tapes associated with our daily backups by employing the IMAGE transaction logging facility. To do so effectively, we must have excellent knowledge and control of our account structures. Transaction logging also delivers many additional benefits. These benefits, and the proper techniques necessary to employ transaction logging - though very relevant to the issue of backup - require a full and separate paper.

Consider, instead of a weekly full, and a daily partial, we employ four different backup job streams:

(1) FULLBUJ, invoked weekly, backs up all non-read-only accounts. The accounts to be backed up are included in an indirect file FULLBUD.DATA.SYSTEM, which is maintained by the system manager, and is FCOPY'd to the $STDLIST of FULLBUJ to ensure it's looked at and properly maintained. Using the account profile examined earlier, the FULLBUD.DATA.SYSTEM file would look something like this:

```
LOG####.PUB.SYS,COMMAND.PUB.SYS
@.@.SYSTEM
@.@.FINANCE
@.@.PERSONEL
@.@.PROD
@.@.TEST
```

(2) PARTBUJ, invoked daily. This backup job stream backs up @.@.@ using the date of the previous full, or partial. If transaction logging is employed, @.@.@ should be replaced with @.@.@-@.DBL.@ to achieve considerable savings.

(3) SYSBUJ, invoked as required. This job backs up only read-only accounts, and is employed whenever necessary to ensure an up-to-date backup of HP and third-party software. The file SYSBUD.DATA.SYSTEM is the indirect file for the STORE (or SYSDUMP) command in this job.

(4) ALLBUJ, invoked as required. This backup is a complete @.@.@, zero date backup, to be used immediately prior to a PLANNED system reload.

It's recommended to use SYSDUMP, instead of STORE, as the backup facility in all cases except SYSBUJ. It's often comforting to have a COLDLOAD tape re-created each day.

This backup strategy can be successfully deployed on well-organized systems with significant benefits.

You should be aware that this method will result in a slightly more complicated recovery procedure - which should not present a problem for the well-run shop. I've always believed that one should strive to optimize your most common events, rather than ones which may never occur.

SUMMARY

A well-planned and organized account structure on the HP3000 can be the springboard for a system that is easy to maintain, can be properly secured and regularly re-secured, provides quicker problem resolution, and - most importantly - optimises the use of valuable human and machine resources.

All it takes is a pinch of technical know-how, meticulous organization, and dose of common-sense.

THE VOICE ON THE LINE

Cherie Schoonover
Hewlett-Packard Company
North American Response Center
3300 Scott Boulevard
Santa Clara, California 95054

The Hewlett-Packard Response Center provides remote telesupport services to HP customers, HP field personnel, and other HP entities. When you call the Response Center, you speak directly with engineer specialists who provide a centralized pool of services available to you. This paper is not intended to give an overview of the services offered through the Response Center but rather to present an outlook of the Response Center from the view of a Response Center Engineer.

A BRIEF HISTORY

Prior to the Response Center (RC), telephone support was available through the local (area/country) level. This support was often referred to as "PICS" -- Phone-In Consulting Service. To prepare for future support needs, localized PICS activities were consolidated into two Response Centers in North America and one Response Center in Europe. Implementation of that plan began in October of 1983 and the Centers opened in February, 1984. Since then, the Response Center's have expanded to include twenty three country Response Centers and Regional Centers in Japan, Australia, Latin America and the Far East.

UNDERSTANDING THE RESPONSE CENTER

The first section of this paper will discuss the internal process involved when you place a call to the Response Center, the resources available to the engineers working on your call, and other activities engineers are involved in. The second section will explore the processes and projects of the Center designed to increase productivity, performance, and the services offered.

Process Flow

By understanding the mechanism which determines what happens to a call from the time it is placed until the time you hear back from an engineer, you will be better prepared to work with the Response Center team in solving your problem.

Placing a Call

A call placed to the Response Center via a toll free 800 number is initially answered by a Call Coordinator. He or she will verify that you have purchased a support contract and that the phone number where you can be reached is correct. They will also "log" the HP application, operating system, or subsystem involved in your question, and a two line description of the problem. After entering/verifying this information, a unique PICS-ID is assigned and the call is sent (via software) to the "Team Leader".

You also have the option of leaving a detailed description of your problem on the voice messaging system. Since you are not limited by the two line problem description, but actually have several minutes for describing your problem, the engineer can often begin working on your problem immediately.

You may want to record the following information:

- The specific model of the computer(s) involved,
- The exact version of the operating system(s),
- The name and version of any software product(s) involved,
- A detailed description of the problem or error that has occurred,
- A description of any recent system or application changes,
- A description of any other system or application abnormalities, even if you think that they are unrelated.

Directing Your Call To a Team

The team leader monitors the incoming calls and assigns them to teams based upon the specialty needed. (Each engineer rates their level of knowledge across technical topics.) By accurately describing your problem and the subsystem involved you can help ensure that your call will be most efficiently routed. Once the calls are assigned they are physically sent to the designated team. The engineers on the team choose to take new calls from the pool based on their expertise and number of open calls. Once an engineer chooses to work on a call they assume ownership of the call. The number of calls handled by an engineer on a daily basis varies greatly based on call loading, engineer expertise, and call difficulty, but is on the average seven calls per day per engineer across all of the product lines supported in the Response Center.

The engineers most important role is that of resource manager. Although a single engineer will assume ownership of a call, calls may in fact be "worked on" by many engineers. The Center works under a "team concept". This means that you have more than just a single engineer working towards a solution for you -- you have a team of specialized engineers with different backgrounds working together to isolate and analyze your problem. Software engineers work closely with hardware and application engineers and vice versa.

One of the special teams your call may be routed to is the System Interrupt Team (SIT). The SIT handles all system "interrupt" calls: System Halts, System Hangs, and System Failures. The engineers on the SIT have special training and have the resources needed to help you quickly recover and bring your system back to operation. They also have access to special diagnostic tools which help analyze the cause of many failures, and in many cases, help prevent reoccurrence.

While the engineer is working on your call they are also responsible for a variety of other calls. For this reason, it is important that you communicate to the engineer the criticality of your problem and the extent of assistance that you need. The engineer works with a variety of customers as well. From novice users to extremely sophisticated users, the engineer strives to deliver the particular support that each customer needs.

Returning Your Call

After the initial call to formulate an accurate problem description, the engineer will usually set up a time to call you back (i.e., if the problem wasn't resolved during the initial call). If it is difficult to reach you it is important that you convey this to the engineer and agree upon a mutually appropriate time for a call back. Likewise, once the problem resolution process has begun, if you want to check on the progress of your call or if you have additional information to pass on to the engineer, you can call back on the 800

number at any time. Callbacks are not made directly to the engineer since they may be on the line with another customer, or away from their phone working on a problem, etc. Instead, by leaving a message for the engineer you are ensured that they will not miss your call nor will you be interrupted when you are on the phone with them. To help avoid "telephone tag" you should be available after placing the call for the return call.

Another mechanism for reducing the frustration of telephone tag is the voice messaging system. When you place a callback you can leave the engineer additional information by requesting to leave a message on the voice mailbox associated with your call.

Tracking Your Call

From the time you place a call through the time that your call is "closed", an online system is used to track the progress of your call. Each time an engineer speaks with you or works on your call, an entry is added to the call "history" (and is available to all engineers and the field) This is extremely important in instances where more than one engineer works on your problem. Other engineers working on your problem can retrieve the existing information to familiarize themselves with your problem, then update it accordingly until a final resolution of your call is reached. Also, the database is used in identifying common problems and as a resource for known-problem solutions.

Once your problem has been resolved, or your call has been referred to your local support team, it will be closed in the tracking system. This should happen as a mutual agreement between you and the engineer working with you. If the problem reoccurs or you find that the workaround suggested is not acceptable after the call is closed, you may open a new call referring back to the original one. Since all closed calls are maintained in the tracking database, we treat your new call as an extension of the old one.

Resources Available to the Engineer

Although some people like to joke of the vision of a Response Center engineer in a public phone booth with roll of dimes in hand, the engineer actually has a wide range of resources available to him or her (including a sophisticated phone system!). Along with the standard HP Manuals and an extensive library, the engineer also has an On-line Data and Information Network system he can use which stores the call documentation on all PICS-calls, Known Problem Reports (KPR's) and Service Requests (SR's), etc. Using keywords (such as "SF311" or "SPOOLEE I/O ERROR") the engineer can search through all the text contained in all the documents for a reference to the keywords.

In conjunction with the online databases, engineers also have a number of people they can use as resources: team members, specialists, online support groups at the divisions, and the local field support teams. Human resources are probably the most useful resources. For example, engineers often contact factory support personnel. With a single phone call the engineer can (and does!) contact the division responsible for a product when further information (not available at the Center) is needed.

Another notable resource is the Diagnostic Center. It contains software and hardware available to the engineer for use in duplicating your system environment. Sometimes referred to as "crash and burn" machines, the engineer is in most cases able to match your system configuration while attempting to duplicate and isolate the true cause of your problem.

The engineer may also choose to dial in to your system. The engineering workstations are set up with a modem on every phone; without leaving his desk, the engineer can access your system as well as expert and diagnostic systems at the Response Center.

Although not yet mentioned, the engineer's most valuable resource is his own knowledge and experience. The experience of engineers ranges from newly trained college-hires to personnel with many years of industry experience.

Engineer Activities

Engineers at the Response Center participate in a number of activities other than solely phone support. From the onset of the Response Center, management realized that engineers would need time to develop and expand their expertise in specialty areas. Thus, the Response Center is staffed to allow for engineers, on the average, to spend a week offline for every two or three weeks online.

Offline activities are scheduled based upon your needs and the needs of the Center coupled with each individuals career and growth interests. By offering a wide range of activities to increase the technical expertise of the engineers, the Response Center attracts highly motivated, technical personnel. Engineer activities include regular phone support, open weeks, projects, classes, preparation for instructing, instructing, field and division exchanges, turnover duty, Service Request (SR) duty, certification, writing Application Notes, and second level support.

Other activities include working on projects such as developing software tools for use by the Response Center or participating in the technical review of a new product or a new release of a product. Engineers also spend a considerable amount of time taking, preparing to instruct, and instructing classes. In conjunction with the instruction of RC-internal classes, engineers also teach customer and factory classes as needed.

The Response Center Training Program is tailored to supplement the engineer's background and specialization interests. In addition to technical training, engineers take courses to help them refine their problem solving skills and professional skills.

Courses have been created to meet the training needs specific to Response Center engineers. One such class is the RCE BASICS course. The goal of RCE BASICS is to develop the professional skills required and to build upon the non-technical knowledge necessary to be successful at the Response Center. The engineers learn how to better manage their workload, their time, and technical and managerial resources to ensure timely and effective resolution of your problems.

Engineers can opt to participate in field or division exchanges. For example, engineers accompany field SE's on account visits or aid the field SE's with onsite assistance. This gives the engineer an opportunity to interface with customers directly and gain direct field experience. Engineers also can spend time working at a division in order to understand the divisions concerns, to form important divisional contacts, and to participate in lab support.

If an engineer is still working on a customer problem at the end of a week, and he or she is not scheduled to be online the following week, then the calls are "turned over" to another engineer. Turnover calls are analyzed and distributed to qualified online engineers. Thus, if you are told by an engineer your call will be turned over, you should be sure to communicate any time sensitivity of your problem. That information can be documented with your call history and the turnover engineer will be better equipped to respond to your needs appropriately.

While working with you on a call, it sometimes becomes necessary for the engineer to submit a Service Request (SR) on your behalf. (An SR is an enhancement request or problem report.) In this situation the RCE assigned to the call usually submits the SR. You can also send SR's directly to the Response Center. It is the responsibility of the engineer assigned to "SR duty" to attempt to duplicate and then submit these SR's.

PRODUCTIVITY, PERFORMANCE, AND SERVICES

For the past three years, the Response Center has provided remote support services to you. In addition, the Response Center support business has been evolving and growing in an effort to recognize and plan for your changing needs and to continually provide you with the highest quality services possible. The remainder of this paper will address some current programs and some programs under investigation; all are intended to increase your productivity, and the productivity, performance, and services of the Response Center.

Patch Project

The goals of the HP 3000 patch project are to provide preventive, centralized, standardized patch distribution, to provide feedback to the divisions, and to provide an information source on the patches themselves. There are three main areas in the program: PATCH ORDERING, PATCH INFORMATION, and ENHANCED CUSTOMER DELIVERABLES. PATCH ORDERING will allow a field SE to call the Response Center to order patches to be delivered directly to you. PATCH INFORMATION services will allow a field SE to request information on any patch. ENHANCED CUSTOMER DELIVERABLES include improved and customized job streams, patch documentation files, and cover letters for those patches sent on tape.

Application Notes

Application Notes are technical articles written by Response Center engineers to help you run your systems more efficiently. HP 3000 Notes are published twice a month and are distributed with the Software Status Bulletin. HP 1000 and HP 9000 notes are published on an as-needed basis. The notes address topics where the volume of calls received at the Center indicates a need for addition to or consolidation of information available through HP support services. The intent is to produce documents which reflect day-to-day customer problems, and are useful to wide ranges of customer interests and expertise. The documents are written so you can easily understand solutions or explanations we have discovered while helping you work with your computer systems.

Unlike a manual which is a technical description of features and functions, the Notes are geared toward explaining concepts, solving problems, and providing information about the use of your HP system on a practical basis. The Notes are supplements to manuals, not replacements for them.

Knowledge Certification/Direct Customer Access

Recently the Centers have begun a program referred to as "Knowledge Certification." Knowledge Certification is the process of creating "Knowledge Abstracts" from known problems and solutions. The purpose of certification is to create a pool of technically verified, consistent, accurate support information. The pool of information will be used to feed the Direct Customer Access program.

Direct Customer Access will provide a fast, easy way for you to communicate with the Response Center and other HP customers concerning support issues. You will be able to connect to Direct Customer Access from your own terminal or workstation via a modem and the X.25 public network. Once logged on, you will be able to:

- Search the Response Center Knowledge Abstracts.
- Exchange private messages with HP or participate in public conversations with other HP customers.
- Access Response Center Questions and Answers or Application Notes.
- Use interactive diagnostic modules to solve system problems.
- Submit problems.

Product Life Cycle

The Response Center is also becoming increasingly involved in the Product Life Cycle. The Center can offer a number of benefits to HP and HP customers by becoming involved in a product prior to its release date. Engineer experts, or "Product Champions", can provide secondary high level support to customers and work with the field in resolving product related problems. The Product Champions can track product information such as call volume and frequency of calls, support issues, product problems, etc. This information can provide the divisions with feedback regarding the product quality and supportability.

Product Support Plans

Product Support Plans (PSPs) define the support model for new products. The goals of Response Center participation in the review of PSPs are: to better our ability to plan our business, and to provide feedback to support planners in the product divisions. In an average month we coordinate the review of 17 new products by 25 managers. We then provide the Response Center's input to monthly Pricing Meetings, where support products are added to the Corporate Price List.

Field Reviews

The Response Center also participates in the field reviews for new operating system/product releases. We make recommendations regarding the quality and supportability of Master Installation Tapes (MITs) to the divisions.

Beta Tests

The purpose of a beta test is to ensure that a product functions according to its specifications and is supportable. The Response Center participates in betas to learn products before they are available to customers, to test support models, and to provide feedback to the divisions.

SE Assist

Along with increasing the range of services offered, the Response Center is also increasing assistance to the field. By making the resources of the Center available to field engineers, ultimately your Account Team will be better able to assist you. An example of field assistance is the SE Assist program.

SE Assist is the process where field engineers contact the Response Center instead of calling directly to the factory. The intent of the program is to leverage off the broad knowledge at the Center and to free up the divisions to work on the problems that require divisional resources. SE Assist was first implemented

with the Commercial Systems Division (MPE support) and currently the feasibility of extending it to other divisions is being investigated.

Software Maintenance Migration

Software support at HP is about to undergo some wide-reaching changes, in which the roles of various field entities, including the Response Center, will be redefined, as will some of the processes for delivering support to you. In preparation for the implementation of these changes the Response Center is helping to redefine CE training and is involved in conducting a test of the our ability to manage onsite field resources in software problem-solving.

CONCLUSION

The Response Center is a key player in future support issues because not only is it the hub for common problem diagnostics, but it is also involved in proactive support services both internally to HP (such as the Product Life Cycle) and externally (such as the Application Notes distributed directly to customers).

Most importantly, the Response Center is an organization dedicated to the future.

ABSTRACT

Garbage In, Garbage Out: Data Integrity Concerns in Multi-User Applications

Eric Schurr, Cognos

The success of an application depends largely on the quality of the data it maintains: in order to extract meaningful information from a system, the data being manipulated must be accurate. Guaranteeing the accuracy of the data in a system extends beyond writing "bug-free" code. Many considerations must be made in terms of locking strategies and concurrency controls if a transaction- based system will be used simultaneously by a number of individuals.

This paper will investigate different strategies that may be used to ensure data integrity in an application. No strategy is optimal for all situations. The strenghs and weaknesses of each approach will be discussed in an attempt to find a strategy that is best-suited for the majority of applications.

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

George B. Scott
ELDEC CORPORATION
16700 13th Avenue West
Lynnwood, WA 98046-0100

I. INTRODUCTION

Response time has become worse. Users are becoming disconcerted. The tension is mounting. You cannot get capital approval for acquisition of additional hardware. You don't have budgeted funds for expensive system performance experts from Hewlett-Packard (HP). Let's face it, you should either start writing you resume or determine how to manage your system resources better.

The emphasis of this paper is on how to obtain and use information from system logfiles to identify and eliminate performance bottlenecks and resource hogs. Special emphasis is given to I-O, CPU usage, process creations and file opens.

Use of information gathered from system logfile analysis will enable you to better understand what resources are being most used; what users, programs and jobs are responsible for that use; and, will give you an overall understanding of what is happening on "your" system.

Following is a outline for the remainder of the paper.

Section	Title
II	What Event Types Can Be Logged
III	Contents of System Logfile Records
IV	Correlation of Data Among Logfile Record Types
V	Summarizing Logfile Record Data
VI	Analysis of Logfile Record Data
VII	Known Limitations/Bugs
VIII	Implementation in Your Shop
IX	Summary
A	Appendix -- Sample Session
B	Appendix -- LISTLOG5 Output for Sample Session
C	Appendix -- FCOPY Dump of Logfile for Sample Session
D	Appendix -- Sample Reports From a Logfile Analysis

II. WHAT EVENT TYPES CAN BE LOGGED

Events shown in Figure 1 may be logged to system logfiles. See your "Systems Operations & Resource Management Reference Manual" for how to turn logging on/off for these events. A complete description of each log record type is contained in Application Note 5, 5/1/86, "MPE System Logfile Record Formats".

Logfile Type	Description
0	Log Failure Record
1	System Up Record
2	Job Initiation Record
3	Job Termination Record
4	Process Termination Record
5	File Close Record
6	System Shutdown Record
7	Power Failure Record
8	Spoolfile Done Record
9	Line Disconnection Record
10	Line Close Record
11	I/O Error Record
12	Physical Mount/Dismount Record
13	Logical Mount/Dismount Record
14	Tape Labels Record
15	Console Log Record
16	Program File Event Record
17	Call Progress Signals Record
18	DCE Provided Info Record
46	MPE Maintenance Request Log Record
47	Diagnostic Control Unit Log Record

Figure 1 -- System Logfile Record Types

III. CONTENTS OF SYSTEM LOGFILE RECORDS

Log Record Header Portion

Each log record has the same format for the first six words as described in Figure 2.

Word	Content
1	Record Type
2	Record Length (Bytes)
3	Date (Calendar Intrinsic Format)
4-5	Time (Clock Intrinsic Format)
6	Job Type/Number (Bits 0-1=Job Type, 2-15=Job Number)

Figure 2 -- Log Record Header Portion

Log Record Detail Portion

The remainder of each record has data specific to the log record type. Record types of particular interest are shown in Figure 3.

Record Type	Field Descriptions
2--Job Init	User, Account, Jobname, Logon Group Name, LDEV Input, Ldev Output, Logon Queue, CPU Limit, Inpri, Outpri
3--Job Term	Max Priority, Num. of Creations, CPU Time in Seconds, Elapsed Time in Minutes
4--Proc Term	Num. of Prog. Segments, Num. of SL Segments, Max Stack Size, Max Data Seg Size, Cumulative Total of Virtual Storage, PIN, CPU Time in Seconds
5--FCLOSE	Filename (File, Group, Account), Disposition, Dev Type, LDEV, Num. of Records, Num. of Blocks
8--Spoolfile	User, Account, Job #, Filename, Spoolfile ID, Device Type, Subtype, Spoolee #, Function, Num. Records Processed, Num. Sectors Used, Num. Pages, #LP/PP
15--Console	Input/Output, Byte Length, Console input or output Line

Figure 3 -- Contents of Selected Logfile Record Types

IV -- CORRELATION OF DATA AMONG LOGFILE RECORD TYPES

Job Related Data

As you can see in Figure 2, each logfile record contains the job number, word six, of the job responsible for the event. Thus you can combine and accumulate information about a job by accumulating records with the same job number.

For example, if a job initiation record is associated with the job termination record, then you can build a single job record such as shown in Figure 4.

Field Description
Job Number
User Name
Account Name
Job Name
Group Name (Logon)
Input LDEV
Output LDEV
Logon Queue
Inpri
Outpri
CPU Time Limit
CPU Time in Seconds
Maximum Priority
Number of Creations
Elapsed Time in Minutes
Date & Time, BOJ
Date & Time, EOJ

Figure 4 -- Job Summary Records, Types 2 & 3

Note: In the above record, the elapsed time could be calculated to the nearest tenth of a second since the BOJ-Time and the EOJ-Time are in the clock intrinsic format.

Now suppose that you associated the File Close (Type 5) log record based on job number. You could count the number of closes, total the number of blocks of I-O of various types, total the number of records of I-O of various types, and create a record such as shown in Figure 5.

Field Description
Job Number
User Name
Account Name
Job Name
Group Name (Logon)
Input LDEV
Output LDEV
Logon Queue
Inpri
Outpri
CPU Time Limit
CPU Time in Seconds
Maximum Priority
Number of Creations
Number of FCLOSES
Blocks of I-O, Disc
" , Tape
" , LP
" , Terminal
" , Misc
Records of I-O,Disc
" ,Tape
" ,LP
" ,Terminal
" ,Misc

Figure 5 -- Job Summary Record, Types 2, 3 & 5

Now suppose that you also gather the number of records that the spooler has processed from Type 8 logfile records. You could then create a record such as shown in Figure 6.

Field Description
Job Number
User Name
Account Name
Job Name
Group Name (Logon)
Input LDEV
Output LDEV
Logon Queue
Inpri
Outpri
CPU Time Limit
CPU Time in Seconds
Maximum Priority
Number of Creations
Number of FCLOSES
Blocks of I-O, Disc
" , Tape
" , LP
" , Terminal
" , Misc
Records of I-O,Disc
" ,Tape
" ,LP
" ,Terminal
" ,Misc
Records, Spooler

Figure 6 -- Job Summary Record, Types 2, 3, 5 & 8

Thus, you can build a single record containing a great deal of information about a job.

The Process Tree

To help understand process related information, let us briefly examine the system process tree. The first process created is PROGEN. Progen creates a number of children, one of which is UCOP, the user controller process. UCOP creates a child process, called a user main process, for each job or session. The user main process is also referred to as the Command Interpreter (CI). When you run a program, it is the child of the CI process. This is summarized graphically in Figure 7.

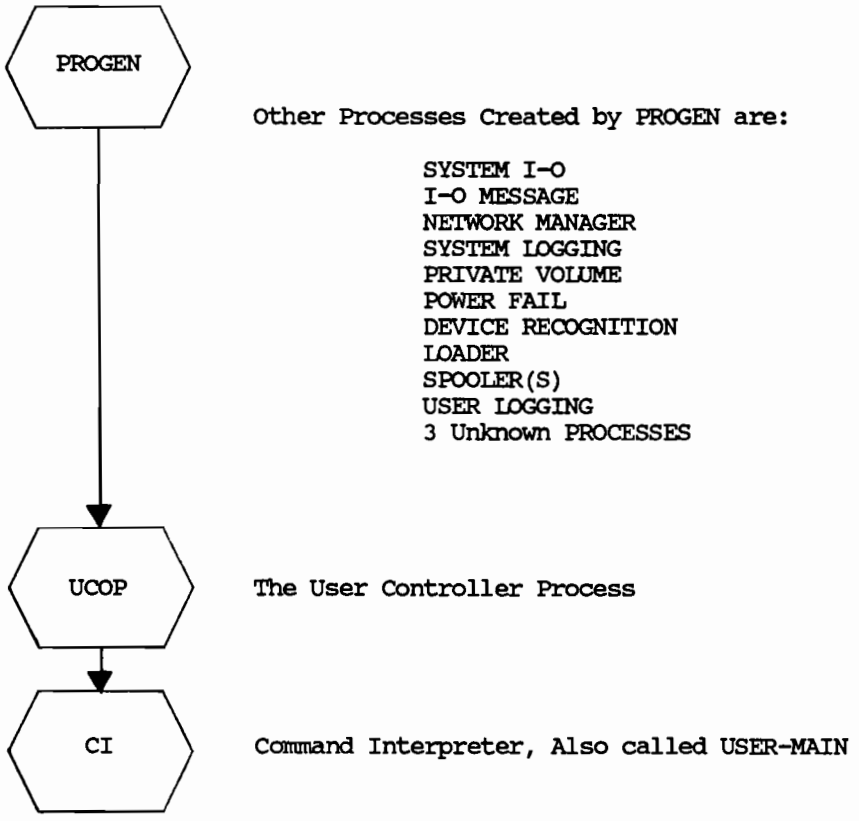


Figure 7 -- Simplified System Process Tree

FIGURE 8 -- Selected Records from APPENDIX B

```
   TIME      TYPE  JOB#        G.02.B0            DATE: SAT, APR, 25,1987     LOGFILE: 5962

15:48:53:3  CONL   S 210    OUT 15:48/#S210/77/LOGON FOR: SCOTT,MGR.TEST,LASVEGAS ON LDEV #30
                    *-----------------------------------------------------------------------------
                      USER    *  ACCOUNT *    JOB     *  LOGON G * LDEV IN  OUT  * RESERVED * CPU LIMIT * INP * OUTP * LOGON Q *
15:48:53:3  JOB    S 210    MGR         TEST        SCOTT       LASVEGAS         30  30     0           -1          8     0       C
                    *-----------------------------------------------------------------------------
                                 FILE NAME          * DISP * DOM *   SECTORS    * DEV T/# *  RECORDS  *   BLOCKS   *
15:48:57:7  FILE   S 210    COMMAND .PUB      .SYS       0      1      251          3  /3      4            4
15:49:16:0  FILE   SYS      TDP     .PUB      .SYS       0      1      710          3  /1      92           92
15:49:24:8  FILE   S 210    TDP     .PUB      .SYS       0      1      710          3  /1      16           16
15:49:25:6  FILE   S 210    TDPPARMS.TDPDATA  .HPOFFICE  0      1      29           3  /4      2            2
15:50:10:2  FILE   S 210    QUERY   .PUB      .SYS       0      1      933          3  /4      94           94
15:50:16:9  FILE   S 210    QSIN    .         .          0      0      0            16 /30     29           29
15:51:24:2  FILE   S 210    $STDINX .         .          0      0      0            16 /30     76           76
15:51:24:3  FILE   S 210    QSMSGCAT.PUB      .SYS       0      1      110          3  /3      16           1
15:51:24:3  FILE   S 210    QSOUT   .         .          1      0      0            16 /30     76           76
15:51:24:3  FILE   S 210    $STDIN  .         .          0      0      0            16 /30     76           76
15:51:24:3  FILE   S 210    $STDLIST.         .          0      0      0            16 /30     76           76
                    *-----------------------------------------------------------------------------
                      PROG SEG * SL SEG * MAX STACK * MAX DS * VIRT ST * PIN #  * CPU TIME *
15:51:24:6  PROC     S 210      16        22       14187      259       1117      113        10
                    *-----------------------------------------------------------------------------
                                 FILE NAME          * DISP * DOM *   SECTORS    * DEV T/# *  RECORDS  *   BLOCKS   *
15:51:44:8  FILE   S 210    $STDIN  .         .          0      0      0            16 /30     88           88
15:51:44:9  FILE   S 210    $STDLIST.         .          0      0      0            16 /30     88           88
                    *-----------------------------------------------------------------------------
                      PROG SEG * SL SEG * MAX STACK * MAX DS * VIRT ST * PIN #  * CPU TIME *
15:51:45:1  PROC     S 210      27        13       4929       132       277       110        3
                    *-----------------------------------------------------------------------------
                                 FILE NAME          * DISP * DOM *   SECTORS    * DEV T/# *  RECORDS  *   BLOCKS   *
15:52:26:1  FILE     SYS      STORE   .PUB      .SYS       0      1      268          3  /1      43           43
15:52:26:3  FILE   S 210    STORE   .PUB      .SYS       0      1      268          3  /1      15           15
                    *-----------------------------------------------------------------------------
15:52:27:9  CONL   S 210    OUT ?15:52/#S210/94/Mount tape of volumeset AAAAAA (ANS)
                    *-----------------------------------------------------------------------------
15:53:27:6  FILE   S 210    $STDIN  .         .          0      0      0            16 /30     110          110
15:53:27:6  FILE   S 210    $STDLIST.         .          0      0      0            16 /30     110          110
                    *-----------------------------------------------------------------------------
                      PROG SEG * SL SEG * MAX STACK * MAX DS * VIRT ST * PIN #  * CPU TIME *
15:53:27:7  PROC     S 210      11        0        12484      68        340       94         2
                    *-----------------------------------------------------------------------------
                      PROG SEG * SL SEG * MAX STACK * MAX DS * VIRT ST * PIN #  * CPU TIME *
15:54:28:5  PROC     S 210      0         0        5808       14        199       77         5
                    *-----------------------------------------------------------------------------
15:54:28:9  CONL   S 210    OUT 15:54/#S210/77/LOGOFF ON LDEV #30
                    *-----------------------------------------------------------------------------
                      MAX PRI * CREAT * CPU TIME(S) * ELAPSED (M) *
15:54:28:9   OFF     S 210      152       3        19             6
                    *-----------------------------------------------------------------------------
                                 FILE NAME          * DISP * DOM *   SECTORS    * DEV T/# *  RECORDS  *   BLOCKS   *
15:54:29:2  FILE   S 210    $STDLIST.         .          0      0      0            16 /30     118          118
15:54:29:4  FILE   S 210    $STDIN  .         .          0      0      0            16 /30     118          118
                    *-----------------------------------------------------------------------------
```

Process Related Information

You may have noticed that the process termination record (Type 4), does not contain the name of the program file which was run. No process initiation record type exists. The file close record doesn't contain the fact that the file being closed was caused by a process initiation. Even with these things going for you, (Thanks, HP!), you can still almost always determine what program is associated with what process termination record. Of course, this takes a little work, has some limited "caveats" and is not "perfect". So, if you are a technical perfectionist, go ahead and start typing your resume because this paper won't save you. If you believe that knowing the programs which are responsible for consumption of 90% of your CPU resources is useful, then continue reading.

To help us demonstrate how one can identify processes within a logfile, a sample session (See Appendix A) was executed. The the LISTLOG5 printout for the resultant logfile is shown in APPENDIX B. To assist in our discussion, the data shown in Figure 8 was extracted from Appendix B.

It is recommended that you follow the sample session along with the resultant log records to familiarize yourself with what is being logged as events take place. To get you started, the following comments are related to the subset of records shown in Figure 8.

A process you can always identify absolutely in the logfiles is the command interpreter process for a job or session. You can observe that at 15:48:53:3, Figure 8 shows the console record for a logon, and shows the PIN number between the session number and the word "LOGON". Noting that the process termination record at 15:54:28:5 in Figure 8 is for the same PIN number, you can thus identify the CI process for the session.

In our example, you then have a file record for COMMAND.PUB.SYS which is associated with your UDC setup.

Next in this example, TDP.PUB.SYS, which was not allocated, was run. Note that two file close records exist for TDP, one by the system and one by S210. TDP then opened the TDPPARMS file.

From within TDP, the allocated program QUERY.PUB.SYS was then run. Note that only one file open occurred and that was by S210. When the QUERY work was finished, (the addition of a record to a long sorted chain), QUERY was exited and the files associated with QUERY, including $STDIN and $STDLIST were closed. The process termination record for QUERY was then posted.

When TDP was exited, the $STDIN and $STDLIST files were closed and the process termination record was then posted for TDP.

A file was then stored. The program STORE.PUB.SYS, which was not allocated at the time, was opened by SYS and then S210. STORE then put a message on the console requesting a tape. Note that the PIN number of the request is 94, which is the PIN number of this execution of STORE. When STORE was ended, the files $STDIN and $STDLIST were closed and the process terminated. Note that the PIN number in this record is also 94 which confirms that this is the PIN for STORE.

The next process termination is for PIN 77 which is the CI for S210. The console message is then logged, followed by the job termination record and then followed by the file close of $STDLIST and $STDIN associated with S210.

Now, let's summarize what has been observed. The logon shows the PIN of the CI. A program that isn't allocated will have two openings of a file, one by SYS and one by the job/session. A file close record for a program has the following characteristics: a disposition of zero, a domain of 1, and an equal number of records and blocks. Thus if you have two consecutive opens, one by SYS and one by the job/session with the other characteristics mentioned, you might have a process starting.

Of course, this doesn't cover the case of allocated programs, and with today's AUTOALLOCATE feature, this is usually the norm. So now you have to go to technique B. First build a list of all programs files on your system. This could be done by listing all files (LISTF) to a file, writing a program to generate a fully qualified program, group and account name for each PROG, BASP, BASFP or any other executable program file code on your system. Then if a FCLOSE log record is for one of these program files, you can generally (>90%) determine that the FCLOSE was the result of a process initiation.

Suppose as you analyze a logfile, you find a console log record for a logon. You save the PIN number and note that this is a CI process initiation. You then may find an FCLOSE record for a program which is in your program file list. If the job is "SYS" you go on. If not, you start a list of program file names for that job/session (perhaps using a data base set). When you come to a process termination record, you check to determine if it is for the CI pin. If not, you do a backward chained on the program list for the job/session of the process termination. The program name found is almost always that of the program which was run. This technique will even work for a process tree where process A creates process B which creates process C since C will terminate prior to B which will terminate prior to A.

Thus, on systems with a relatively stable set of programs, the above technique delivers generally accurate (>90%) results.

Now for some reasons why it is not "perfect". If you have a process tree with multiple programs files on a single level which are different programs or multiple levels with multiple programs running on each level, the order of termination of the program files will probably not be the reverse of the order of creation (LIFO).

Ask yourself how much of your CPU is being consumed by such a process tree. Sometimes you can use your own knowledge to determine the true program names where the names may have been switched between two or three processes. Generally, this is better than having no insight at all.

Another way that this technique may be misleading is to build a program file while running a user program. The last program in the job chain would then be the file built, not the one executed. You have to ask yourself how many times you do this on your system, (Once a year, perhaps?).

You are generally safe when compiling programs. The object code from the compiler starts as $NEWPASS which becomes $OLDPASS. Thus the process termination is correctly associated with the compiler. The object code from the prep ($OLDPASS) is closed with a disposition of 2. When the process termination record is logged, the process can then be associated with SEGPROC. The program file with a disposition of one is logged AFTER the process termination record for SEGPROC. Of course, a entry will be added to your job chain for this program, but it will never be popped off because no active processes exist to create a process termination record for this program/job.

To summarize, you can usually correlate the program file which has been closed (Type 5) to the process termination record (Type 4). From this, you can build a process data record such as shown in Figure 9.

Source	Field Descriptions
5--FCLOSE	Program Name (Name, Group, Account)
"	Date-Time Started
4--Proc Term	Date-Time Ended
	HPJOB-NUM
2--Job Init	Jobname
4--Proc Term	PIN
	CPU in Seconds
"	Max. Stack Size
"	Num. of Program File Segments
"	Num. of SL Segments
"	Max. Data Seg. Ever
"	Cumulative Total Virtual Storage

Figure 9 -- Correlated Process Termination Record

V. SUMMARIZING LOGFILE RECORD DATA

If you create three basic records: a job record such as shown in Figure 6; a process record such as shown in Figure 9; and, a file close record such as shown in Figure 10; you can then summarize the data to create reports by account, by device, by filename and by program name. See Appendix D for sample reports which were generated from a system logfile analysis.

Field Description
Date-Time FCLOSE
Job Number
Filename (Name,Group,Account)
Disposition
Domain
Sectors
Device Type
I-O, Records
I-O, Blocks
LDEV

Figure 10 -- File Close Record

VI. ANALYSIS OF LOGFILE RECORD DATA

This section deals with a discussion on various methods you might want to use to gain insight into system performance problems using data generated by logfile analysis. This is not meant to be a tutorial on system performance, but simply some thoughts on where to start. Simply stated, if you have identified who or what is consuming your resources, you need only to fix the problem. Examples of the reports mentioned below are shown in Appendix D.

I-O Analysis

Two approaches are commonly used to identify the source of I-O problems from system logfile analysis. The first is to determine which job is doing the greatest amount of I-O. This can be determined from a report such as RJOBSIO using job summary information. After determining which job is using the greatest amount of I-O, then you can determine which files are involved by finding all files which have I-O for the job number in question. These results can be reported using a report such as RT05, which shows file close information.

The second approach is to first determine which file is being accessed most frequently from a report such as the RFILESIO, which reports file summary data. Next, determine which jobs accessed the most used file from the RT05 report and finally, determine the jobname, account, etc. from the job summary data.

Both approaches are similar in ease of analysis. It is quite common to find that the biggest culprit of I-O use is a result of using a poor blocking factor for a file.

CPU Analysis

Similar to I-O analysis, two main approaches exist for determining the source of CPU use. The first is to determine which job is using the greatest amount of CPU by reviewing the RJOBSCPU report. Next, using the job number, find all processes for that job in the RPROCS report.

The second approach is to analyze the RPROCSUM report first to find which programs are consuming the greatest amount of CPU and then use the RPROCS report to determine which jobs are executing the program in question.

Again, each approach is similar in ease of use.

Process Creation Analysis

A very costly operation is the creation of a process. This happens most often when a user enters the "RUN" command. For sites using process handling, each time a process is created or terminated causes a costly operation with respect to CPU use. One way to analyze this is to determine which jobs have the greatest number of process creations (NUM-CREATIONS) by analyzing the RPROCSUM report.

A second approach would be to just count the number of creations for a program in the RPROCS report and then determine what jobs are causing these creations.

FOPEN/FCLOSE Analysis

The total number of FCLOSE's executed for each job is contained in the RJOBSIO report. After determining which job(s) is the culprit, one can then find the file names of the jobs being accessed by that job in the RT05 report.

A second approach is to use the RFILESIO report to see which files have a large number of closes and then backtrack to determine which jobs are causing this to occur.

Analysis Summary

The preceding comments are neither comprehensive nor exhaustive. They are merely to show you that you can not only determine where the bulk of your resources are being consumed, but that you can also determine what program and/or files are at the root of the problem. See Appendix D for an example of a very simplified analysis.

As you become more proficient with logfile analysis, you will be able to spot problems caused by data base structures, blocking factors, UDC's, etc.

VII. KNOWN LIMITATIONS / BUGS

Logfile analysis depends on Hewlett-Packard's system logging to be accurate. Any error or bug in the system logging by MPE will obviously be carried over into the resulting reports and data. At this point, one known problem which may bias results in a few rare cases exists for those sites which explicitly use GMULTI access for files. As of UB-MIT for MPE 5E, the system logs the total I-O done be all users at the time when each user closes a file. Two methods exist to correct this problem. First, you can adjust the I-O correctly by changing the counts in your data. Secondly, you can change the programs to not open files in GMULTI mode. Note that this causes no problem for anyone not deliberately opening files in GMULTI mode and having several processes accessing the file. Note also, that unless the quantity of I-O to such a file is large, the overall results are unlikely to be biased.

A second known problem is that MPE is occasionally not logging the correct device number for FCLOSE. This is somewhat more rare but does cause the dev type to not always correspond correctly to the device number.

One should not be frightened by these aberrations. The great wealth of information available from system logfile analysis greatly exceeds the small amount of inaccuracies. Please note that other products such as OPT/3000, an Hewlett-Packard product, also have bugs. In fact, if you review the Software Status Bulletin (SSB), you will find that most of the products have errors in them; however, this does not impede thousands of companies from successfully using MPE and the HP3000 to support their business and technical operations.

VIII. IMPLEMENTING LOGFILE ANALYSIS IN YOUR SHOP

There are a variety of ways to implement logfile analysis in your shop. You might use HP's LISTLOG5.PUB.SYS, or use a contributed library program, or purchase some very reasonable priced logfile analysis tool, or write a utility yourself.

The source doesn't really matter that much. To answer a very simple question, LISTLOG5 might suffice. To do a comprehensive analysis, you should have a tool which can produce several reports and perhaps load the data into a data base for ad hoc analysis.

Hopefully, the following suggestions will make your entry into this arena a little less rugged.

1. Try and select a period which represents a significant and uniform activity type. For example, break your workday into two periods from 7AM to 6PM and from 6PM to 7AM. Analyze these periods as an on-line heavy user period vs. a nighttime batch oriented period.

2. To get the best results, try and create a period of quiescence, when no jobs or sessions are running, at the beginning and end of your analysis period. During this time of quiescence, do a SWITCHLOG.

3. Perform such an analysis for several days. You will quickly perceive what is a pattern of usage and what is probably a one time only occurrence. If you contrast this technique to a one time only shot of performance analysis which covers a one or two hour "typical" period, you will readily discover which gives you a more comprehensive understanding of your system.

IX. SUMMARY

If you've made it this far, then you're probably serious about trying to improve your performance using system logfile analysis. You have seen that system logfiles are a good source of information about what is happening on your system. They are particularly valuable in providing you with totals for I-O, CPU, FCLOSES, process creations, spooler activity, etc. over a selected period of time.

Other tools and methods of analysis often give you insight as to what is happening at a particular point in time. System logfile analysis can be used to complement those tools and not leave you wondering whether the event you observed was typical or whether it was a unique or rare event.

By using system logfile analysis repeatedly, you will usually discover some resource hog or performance bottleneck which was totally unsuspected at the start of the analysis. You will also be able to keep your system relatively clean from those little problems such as inappropriately blocked files which are incurring 100,000 I-O per day.

Good Luck.

APPENDIX A

Appendix A contains a sample session which was used to generate a logfile (5962) to demonstrate the relationship between events you as a user cause and the logfile records that are generated to document these events. Several files were built during this session to merely provide points of reference for you in the logfile. See the subsection on "Process Related Information" in Section IV for a discussion and a simplified analysis.

APPENDIX A

```
:HELLO SCOTT,MGR.TEST,LASVEGAS
SYSD HP3000/MPE V  G.02.01  (BASE  G.02.01).   SAT, APR 25, 1987,  3:48 PM
:BUILD LS1
:FILE LP;DEV=LP1,13,1
:RUN TDP.PUB.SYS
TDP/3000    (A.04.01)   HP36578 Editor (c) COPYRIGHT Hewlett-Packard Co. 1986
          SAT, APR 25, 1987   3:49 PM   (DAY #115)
/A 1
     1      THIS IS A TEST
     2      //
/K LS2,UNN
/RUN QUERY.PUB.SYS
HP32216C.00.03 QUERY/3000   SAT, APR 25, 1987,   3:50 PM
COPYRIGHT HEWLETT-PACKARD CO. 1976

>X DBSECON.DATADP.ELDEC

   executes logon xeq file


END OF XEQ FILE
>ADD COMD-GROUPS

   add entry to set comd-groups

>EXIT
 END OF PROGRAM

TDP/3000
/K LS3,UNN
/L ALL,OFFLINE
Listing offline **
/BUILD LS4
/EXIT

END OF PROGRAM
:BUILD LS5
:FILE TAPE;DEV=TAPE;LABEL=AAAAAA
:STORE LS1;*TAPE;SHOW
STORE/RESTORE, VERSION 2     (C) 1981  HEWLETT-PACKARD CO.
SAT, APR 25, 1987,  3:52 PM

FILENAME.GROUP   .ACCOUNT   LDN  ADDRESS REEL   SECTORS CODE

LS1      .LASVEGAS.TEST        3%00667401    1       128

FILES STORED:          1
:BUILD LD6
:TELLOP;LAS VEGAS DEMO DONE
:BYE

SYSD CPU=19. CONNECT=6. SAT, APR 25, 1987,  3:54 PM
```

APPENDIX B

Appendix B contains output from LISTLOG5.PUB.SYS for logfile 5962.

```
   TIME     TYPE  JOB#       G.02.B0            DATE: SAT, APR, 25,1987     LOGFILE: 5962

                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:48:38:3  FILE  SYS     LOG5961 .PUB      .SYS       0     1     128         3  /2     50          6
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:48:38:6  FILE  SYS     LOG5962 .PUB      .SYS       1     0     128         3  /2     0           0
15:48:38:7  CONL  SYS     OUT LOG FILE NUMBER 5962 ON
15:48:53:3  CONL  S 210   OUT 15:48/#S210/77/LOGON FOR: SCOTT,MGR.TEST,LASVEGAS ON LDEV #30
                   USER    *  ACCOUNT *   JOB     *  LOGON G * LDEV IN  OUT  * RESERVED * CPU LIMIT * INP * OUTP * LOGON Q *
15:48:53:3  JOB   S 210   MGR          TEST       SCOTT       LASVEGAS       30  30     0           -1          8     0       C
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:48:57:7  FILE  S 210   COMMAND .PUB      .SYS       0     1     251         3  /3     4           4
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:49:12:1  FILE  S 210   LS1     .LASVEGAS.TEST       1     0     128         3  /3     0           0
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:49:16:0  FILE  SYS     TDP     .PUB      .SYS       0     1     710         3  /1     92          92
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:49:24:8  FILE  S 210   TDP     .PUB      .SYS       0     1     710         3  /1     16          16
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:49:25:6  FILE  S 210   TDPPARMS.TDPDATA .HPOFFICE    0     1     29          3  /4     2           2
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:49:48:2  FILE  S 210   K1151549.LASVEGAS.TEST       1     0     1008        3  /4     0           0
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:50:6 :1  FILE  S 210   LS2     .LASVEGAS.TEST       1     0     2           3  /1     1           1
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:50:10:2  FILE  S 210   QUERY   .PUB      .SYS       0     1     933         3  /4     94          94
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:50:16:9  FILE  S 210   QSIN    .         .          0     0     0           16 /30    29          29
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:50:17:6  FILE  S 210   TDACAT  .PUB      .SYS       0     1     165         3  /4     0           0
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:50:19:8  FILE  S 210   DBSECON .DATADP   .ELDEC     0     1     6           3  /2     8           1
                                 FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:51:23:4  FILE  S 210   DBSECQ  .DATADP   .ELDEC     0     1     127         3  /3     109         109
```

APPENDIX B

TIME TYPE JOB# G.02.B0 DATE: SAT, APR, 25,1987 LOGFILE: 5962

TIME	TYPE	JOB#	FILE NAME	DISP	DOM	SECTORS	DEV T/#	RECORDS	BLOCKS
15:51:23:5	FILE	S 210	QSKIB .LASVEGAS.TEST	0	0	4	3 /3	0	0
15:51:23:6	FILE	S 210	DBSEC04 .DATADP .ELDEC	0	1	1096	3 /2	0	0
15:51:23:6	FILE	S 210	DBSEC04 .DATADP .ELDEC	0	1	1096	3 /2	1	1
15:51:23:6	FILE	S 210	DBSEC04 .DATADP .ELDEC	0	1	1096	3 /2	1	1
15:51:23:7	FILE	S 210	DBSEC06 .DATADP .ELDEC	0	1	1323	3 /4	0	0
15:51:23:7	FILE	S 210	DBSEC06 .DATADP .ELDEC	0	1	1323	3 /4	1	1
15:51:23:8	FILE	S 210	DBSEC06 .DATADP .ELDEC	0	1	1323	3 /4	1	1
15:51:23:8	FILE	S 210	DBSEC11 .DATADP .ELDEC	0	1	954	3 /1	0	0
15:51:23:8	FILE	S 210	DBSEC11 .DATADP .ELDEC	0	1	954	3 /1	1	1
15:51:23:9	FILE	S 210	DBSEC11 .DATADP .ELDEC	0	1	954	3 /1	1	1
15:51:23:9	FILE	S 210	DBSEC12 .DATADP .ELDEC	0	1	1632	3 /2	0	0
15:51:24:0	FILE	S 210	DBSEC12 .DATADP .ELDEC	0	1	1632	3 /2	1	1
15:51:24:0	FILE	S 210	DBSEC12 .DATADP .ELDEC	0	1	1632	3 /2	1	1
15:51:24:0	FILE	S 210	DBSEC19 .DATADP .ELDEC	0	1	2968	3 /1	0	0
15:51:24:1	FILE	S 210	DBSEC19 .DATADP .ELDEC	0	1	2968	3 /1	5	5
15:51:24:1	FILE	S 210	DBSEC19 .DATADP .ELDEC	0	1	2968	3 /1	351	351
15:51:24:2	FILE	S 210	DBSEC .DATADP .ELDEC	0	1	32	3 /2	23	23

```
   TIME     TYPE  JOB#       G.02.B0          DATE: SAT, APR, 25,1987    LOGFILE: 5962
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:24:2  FILE  S 210  $STDINX .     .       0     0     0          16 /30    76          76
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:24:3  FILE  S 210  QSMSGCAT.PUB   .SYS    0     1     110        3  /3     16          1
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:24:3  FILE  S 210  QSOUT   .     .       1     0     0          16 /30    76          76
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:24:3  FILE  S 210  $STDIN  .     .       0     0     0          16 /30    76          76
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:24:3  FILE  S 210  $STDLIST.     .       0     0     0          16 /30    76          76
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:24:4  FILE  S 210       .LASVEGAS.TEST    0     0     6          3  /2     0           0
*-----------------------------------------------------------------------------------------*
                  PROG SEG * SL SEG * MAX STACK * MAX DS * VIRT ST * PIN #  * CPU TIME *
15:51:24:6  PROC  S 210        16        22       14187      259       1117      113        10
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:32:3  FILE  S 210  LS3       .LASVEGAS.TEST    1     0     2          3  /4     1           1
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:36:1  FILE  S 210  LP        .LASVEGAS.TEST    0     0     32         32 /1     4           1
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:36:6  FILE  SYS    TT18    .PUB     .SYS    0     1     2          3  /1     2           1
*-----------------------------------------------------------------------------------------*
                   USER  *  ACCT  *  JOBN  *  FILE  *  J/S  *I/O*DEVID*DEV*SP#*COP*PRI*
15:51:36:7  SPOO  SYS    MGR       TEST      SCOTT     LP       S210     OUT 1377  32  24   0   13
                    RECORDS *  SECTORS *FUNC*  PAGES   *LP/PP*SUBTYPE
                   6          32         0                  14
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:36:9  FILE  SYS       .LASVEGAS.TEST    4     1     32         3  /1     4           1
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:42:1  FILE  S 210  LS4       .LASVEGAS.TEST    1     0     128        3  /1     0           0
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:44:8  FILE  S 210  K1151549.LASVEGAS.TEST    4     1     1008       3  /4     20          3
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:44:8  FILE  S 210  $STDIN  .     .       0     0     0          16 /30    88          88
*-----------------------------------------------------------------------------------------*
                        FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *  BLOCKS  *
15:51:44:9  FILE  S 210  $STDLIST.     .       0     0     0          16 /30    88          88
*-----------------------------------------------------------------------------------------*
```

```
TIME        TYPE  JOB#        G.02.B0            DATE: SAT, APR, 25,1987      LOGFILE: 5962
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:51:44:9  FILE  S 210    $STDINX .        .            0      0      0            16 /30    88              88
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:51:44:9  FILE  S 210    $STDLIST.        .            0      0      0            16 /30    88              88
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                           PROG SEG * SL SEG * MAX STACK * MAX DS * VIRT ST * PIN #  * CPU TIME *
15:51:45:1  PROC  S 210     27         13        4929        132       277      110        3
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:51:51:9  FILE  S 210    LS5     .LASVEGAS.TEST        1      0      128          3  /2     0               0
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:52:26:1  FILE  SYS      STORE   .PUB      .SYS        0      1      268          3  /1     43              43
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:52:26:3  FILE  S 210    STORE   .PUB      .SYS        0      1      268          3  /1     15              15
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                          LDEV# FILE# FILE SEQ# SEQ TY TYPE PIN# VOL# EXP DATE FILENAME         LOCKWORD VOLSET ID VOL ID
15:52:27:7  TAPL  S 210    0     6      1         1     ANSI 94    1   000000   TAPE    .                 AAAAAA    AAAAAA
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
15:52:27:9  CONL  S 210    OUT ?15:52/#S210/94/Mount tape of volumeset AAAAAA (ANS)
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
15:53:12:9  CONL  SYS      OUT 15:53/11/Reel 1, vol AAAAAA (ANS) mounted on LDEV# 7
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:53:22:4  FILE  S 210    GOOD    .LASVEGAS.TEST        0      0      32           3  /3     5               5
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:53:22:4  FILE  S 210    SYSLIST .        .            0      0      0·           16 /30    110             110
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:53:27:5  FILE  S 210    TAPE    .        .            1      0      0            24 /7     2               1
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:53:27:6  FILE  S 210            .LASVEGAS.TEST        0      0      17           3  /4     0               0
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:53:27:6  FILE  S 210    $STDIN  .        .            0      0      0            16 /30    110             110
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:53:27:6  FILE  S 210    $STDLIST.        .            0      0      0            16 /30    110             110
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                           PROG SEG * SL SEG * MAX STACK * MAX DS * VIRT ST * PIN #  * CPU TIME *
15:53:27:7  PROC  S 210     11         0         12484       68        340      94         2
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                                   FILE NAME          * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:54:16:3  FILE  S 210    LD6     .LASVEGAS.TEST        1      0      128          3  /1     0               0
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
15:54:26:7  CONL  S 210    OUT 15:54/#S210/77/FROM/MGR.TEST/;LAS VEGAS DEMO DONE
                          * . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

APPENDIX B

```
   TIME      TYPE  JOB#        G.02.B0            DATE: SAT, APR, 25,1987     LOGFILE: 5962

                          *-----------------------------------------------------------------------------
                           PROG SEG * SL SEG * MAX STACK * MAX DS * VIRT ST * PIN #  * CPU TIME *
                                 FILE NAME            * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:54:28:5  FILE  S 210    DP002C  .DATADP  .TEST        0     1     95          3  /4     286          18
                          *-----------------------------------------------------------------------------
                                 FILE NAME            * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:54:28:5  FILE  S 210    DP002C  .DATADP  .SYS         0     1     200         3  /2     617          39
                          *-----------------------------------------------------------------------------
15:54:28:5  PROC  S 210      0         0        5808        14        199      77        5
                          *-----------------------------------------------------------------------------
15:54:28:9  CONL  S 210    OUT 15:54/#S210/77/LOGOFF ON LDEV #30
                          *-----------------------------------------------------------------------------
                           MAX PRI * CREAT * CPU TIME(S) * ELAPSED (M) *
15:54:28:9   OFF  S 210      152     3       19            6
                          *-----------------------------------------------------------------------------
                                 FILE NAME            * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:54:29:2  FILE  S 210    $STDLIST.        .           0     0     0           16 /30     118          118
                          *-----------------------------------------------------------------------------
                                 FILE NAME            * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:54:29:4  FILE  S 210    $STDIN   .       .           0     0     0           16 /30     118          118
                          *-----------------------------------------------------------------------------
                                 FILE NAME            * DISP * DOM *  SECTORS   * DEV T/# *  RECORDS  *   BLOCKS   *
15:54:29:4  FILE  SYS      TT10     .PUB     .SYS        0     1     2           3  /1     2            1
                          *-----------------------------------------------------------------------------
```

APPENDIX C

Appendix C contains a character and octal listing of logfile 5962 produced by FCOPY.

```
LOG5962 RECORD 0 (%0, #0)
00000: 000005 000035 127163 007460 023003 000000 046117 043465 034466 030440 027120 052502 .....s.0&...LOG5961 .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 000176 000001 000000 000200 001400      .SYS     .~........
00030: 000000 000062 000000 000006 000002                                                    ...2......

LOG5962 RECORD 1 (%1, #1)
00000: 000005 000035 127163 007460 023006 000000 046117 043465 034466 031040 027120 052502 .....s.0&...LOG5962 .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 000176 000400 000000 000200 001400      .SYS     .~........
00030: 000000 000000 000000 000000 000002                                                    ..........

LOG5962 RECORD 2 (%2, #2)
00000: 000017 000023 127163 007460 023007 000000 000027 046117 043440 043111 046105 020116 .....s.0&.....LOG FILE N
00014: 052515 041105 051040 032471 033062 020117 047347                                      UMBER 5962 ON.

LOG5962 RECORD 3 (%3, #3)
00000: 000017 000046 127163 007460 032403 040322 000075 030465 035064 034057 021523 031061 ...&.s.05.@..=15:48/#S21
00014: 030057 033467 027514 047507 047516 020106 047522 035040 051503 047524 052054 046507 0/77/LOGON FOR: SCOTT,MG
00030: 051056 052105 051524 026114 040523 053105 043501 051440 047516 020114 042105 053040 R.TEST,LASVEGAS ON LDEV
00044: 021463 030011                                                                         #30.

LOG5962 RECORD 4 (%4, #4)
00000: 000002 000036 127163 007460 032403 040322 046507 051040 020040 020040 052105 051524 .....s.05.@.MGR     TEST
00014: 020040 020040 051503 047524 052040 020040 046101 051526 042507 040523 000036 000036     SCOTT   LASVEGAS....
00030: 000000 000103 177777 177777 004000 000000                                            ...C........

LOG5962 RECORD 5 (%5, #5)
00000: 000005 000035 127163 007460 034407 040322 041517 046515 040516 042040 027120 052502 .....s.09.@.COMMAND .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 050000 000001 000000 000373 001400      .SYS     P.........
00030: 000000 000004 000000 000004 000003                                                    ..........

LOG5962 RECORD 6 (%6, #6)
00000: 000005 000035 127163 007461 006001 040322 046123 030440 020040 020040 027114 040523 .....s.1..@.LS1      .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 005400 000400 000000 000200 001400 VEGAS.TEST     ..........
00030: 000000 000000 000000 000000 000003                                                    ..........

LOG5962 RECORD 7 (%7, #7)
00000: 000005 000035 127163 007461 010000 000000 052104 050040 020040 020040 027120 052502 .....s.1....TDP      .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 000374 000001 000000 001306 001400      .SYS     ..........
00030: 000000 000134 000000 000134 000001                                                    ...\...\..

LOG5962 RECORD 8 (%10, #8)
00000: 000005 000035 127163 007461 014010 040322 052104 050040 020040 020040 027120 052502 .....s.1..@.TDP      .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 001121 000001 000000 001306 001400      .SYS     .Q........
00030: 000000 000020 000000 000020 000001                                                    ..........

LOG5962 RECORD 9 (%11, #9)
00000: 000005 000035 127163 007461 014406 040322 052104 050120 040522 046523 027124 042120 .....s.1..@.TDPPARMS.TDP
00014: 042101 052101 020056 044120 047506 043111 041505 000000 000001 000000 000035 001400 DATA .HPOFFICE..........
00030: 000000 000002 000000 000002 000004                                                    ..........
```

```
LOG5962 RECORD 10 (%12, #A)
00000: 000005 000035 127163 007461 030002 040322 045461 030465 030465 032071 027114 040523 .....s.10.a.K1151549.LAS
00014: 053105 043501 051456 052105 051524 020040 020040 000000 000400 000000 001760 001400 VEGAS.TEST  ..........
00030: 000000 000000 000000 000000 000004                                                    ..........

LOG5962 RECORD 11 (%13, #B)
00000: 000005 000035 127163 007462 003001 040322 046123 031040 020040 020040 027114 040523 .....s.2..a.LS2    .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 000406 000400 000000 000002 001400 VEGAS.TEST  ..........
00030: 000000 000001 000000 000001 000001                                                    ..........

LOG5962 RECORD 12 (%14, #C)
00000: 000005 000035 127163 007462 005002 040322 050525 042522 054440 020040 027120 052502 .....s.2..a.QUERY  .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 000406 000001 000000 001645 001400     .SYS    ..........
00030: 000000 000136 000000 000136 000004                                                    ..........

LOG5962 RECORD 13 (%15, #D)
00000: 000005 000035 127163 007462 010011 040322 050523 044516 020040 020040 027040 020040 .....s.2..a.QSIN   .
00014: 020040 020040 020056 020040 020040 020040 020040 010105 000000 000000 000000 010000     .         .E.......
00030: 000000 000035 000000 000035 000036                                                    ..........

LOG5962 RECORD 14 (%16, #E)
00000: 000005 000035 127163 007462 010406 040322 052104 040503 040524 020040 027120 052502 .....s.2..a.TDACAT  .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 000105 000001 000000 000245 001400     .SYS    .E.......
00030: 000000 000000 000000 000000 000004                                                    ..........

LOG5962 RECORD 15 (%17, #F)
00000: 000005 000035 127163 007462 011410 040322 042102 051505 041517 047040 027104 040524 .....s.2..a.DBSECON .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 001400 000001 000000 000006 001400 ADP  .ELDEC  ..........
00030: 000000 000010 000000 000001 000002                                                    ..........

LOG5962 RECORD 16 (%20, #10)
00000: 000005 000035 127163 007463 013404 040322 042102 051505 041521 020040 027104 040524 .....s.3..a.DBSECQ  .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 000105 000001 000000 000177 001400 ADP  .ELDEC  .E.......
00030: 000000 000155 000000 000155 000003                                                    ...m...m..

LOG5962 RECORD 17 (%21, #11)
00000: 000005 000035 127163 007463 013405 040322 050523 045511 041040 020040 027114 040523 .....s.3..a.QSKIB   .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 000105 000000 000000 000004 001400 VEGAS.TEST  .E.......
00030: 000000 000000 000000 000000 000003                                                    ..........

LOG5962 RECORD 18 (%22, #12)
00000: 000005 000035 127163 007463 013406 040322 042102 051505 041460 032040 027104 040524 .....s.3..a.DBSEC04 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 002110 001400 ADP  .ELDEC  CE.....H..
00030: 000000 000000 000000 000000 000002                                                    ..........

LOG5962 RECORD 19 (%23, #13)
00000: 000005 000035 127163 007463 013406 040322 042102 051505 041460 032040 027104 040524 .....s.3..a.DBSEC04 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 002110 001400 ADP  .ELDEC  CE.....H..
00030: 000000 000001 000000 000001 000002                                                    ..........
```

APPENDIX C

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

APPENDIX C

```
LOG5962 RECORD 20 (%24, #14)
00000: 000005 000035 127163 007463 013406 040322 042102 051505 041460 032040 027104 040524 .....s.3..@.DBSEC04 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 002110 001400 ADP  .ELDEC   CE.....H..
00030: 000000 000001 000000 000001 000002                                                   ..........

LOG5962 RECORD 21 (%25, #15)
00000: 000005 000035 127163 007463 013407 040322 042102 051505 041460 033040 027104 040524 .....s.3..@.DBSEC06 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 002453 001400 ADP  .ELDEC   CE.....+..
00030: 000000 000000 000000 000000 000004                                                   ..........

LOG5962 RECORD 22 (%26, #16)
00000: 000005 000035 127163 007463 013407 040322 042102 051505 041460 033040 027104 040524 .....s.3..@.DBSEC06 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 002453 001400 ADP  .ELDEC   CE.....+..
00030: 000000 000001 000000 000001 000004                                                   ..........

LOG5962 RECORD 23 (%27, #17)
00000: 000005 000035 127163 007463 013410 040322 042102 051505 041460 033040 027104 040524 .....s.3..@.DBSEC06 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 002453 001400 ADP  .ELDEC   CE.....+..
00030: 000000 000001 000000 000001 000004                                                   ..........

LOG5962 RECORD 24 (%30, #18)
00000: 000005 000035 127163 007463 013410 040322 042102 051505 041461 030440 027104 040524 .....s.3..@.DBSEC11 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 001672 001400 ADP  .ELDEC   CE........
00030: 000000 000000 000000 000000 000001                                                   ..........

LOG5962 RECORD 25 (%31, #19)
00000: 000005 000035 127163 007463 013410 040322 042102 051505 041461 030440 027104 040524 .....s.3..@.DBSEC11 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 001672 001400 ADP  .ELDEC   CE........
00030: 000000 000001 000000 000001 000001                                                   ..........

LOG5962 RECORD 26 (%32, #1A)
00000: 000005 000035 127163 007463 013411 040322 042102 051505 041461 030440 027104 040524 .....s.3..@.DBSEC11 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 001672 001400 ADP  .ELDEC   CE........
00030: 000000 000001 000000 000001 000001                                                   ..........

LOG5962 RECORD 27 (%33, #1B)
00000: 000005 000035 127163 007463 013411 040322 042102 051505 041461 031040 027104 040524 .....s.3..@.DBSEC12 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 003140 001400 ADP  .ELDEC   CE.....'..
00030: 000000 000000 000000 000000 000002                                                   ..........

LOG5962 RECORD 28 (%34, #1C)
00000: 000005 000035 127163 007463 014000 040322 042102 051505 041461 031040 027104 040524 .....s.3..@.DBSEC12 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 003140 001400 ADP  .ELDEC   CE.....'..
00030: 000000 000001 000000 000001 000002                                                   ..........

LOG5962 RECORD 29 (%35, #1D)
00000: 000005 000035 127163 007463 014000 040322 042102 051505 041461 031040 027104 040524 .....s.3..@.DBSEC12 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 003140 001400 ADP  .ELDEC   CE.....'..
00030: 000000 000001 000000 000001 000002                                                   ..........
```

```
LOG5962 RECORD 30 (%36, #1E)
00000: 000005 000035 127163 007463 014000 040322 042102 051505 041461 034440 027104 040524 .....s.3..@.DBSEC19 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 005630 001400 ADP  .ELDEC   CE........
00030: 000000 000000 000000 000000 000001                                                   ..........

LOG5962 RECORD 31 (%37, #1F)
00000: 000005 000035 127163 007463 014001 040322 042102 051505 041461 034440 027104 040524 .....s.3..@.DBSEC19 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 005630 001400 ADP  .ELDEC   CE........
00030: 000000 000005 000000 000005 000001                                                   ..........

LOG5962 RECORD 32 (%40, #20)
00000: 000005 000035 127163 007463 014001 040322 042102 051505 041461 034440 027104 040524 .....s.3..@.DBSEC19 .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 041505 000001 000000 005630 001400 ADP  .ELDEC   CE........
00030: 000000 000537 000000 000537 000001                                                   ..._..._..

LOG5962 RECORD 33 (%41, #21)
00000: 000005 000035 127163 007463 014002 040322 042102 051505 041440 020040 027104 040524 .....s.3..@.DBSEC   .DAT
00014: 040504 050040 020056 042514 042105 041440 020040 177505 000001 000000 000040 001400 ADP  .ELDEC   .E..... ..
00030: 000000 000027 000000 000027 000002                                                   ..........

LOG5962 RECORD 34 (%42, #22)
00000: 000005 000035 127163 007463 014002 040322 022123 052104 044516 054040 027040 020040 .....s.3..@.$STDINX .
00014: 020040 020040 020056 020040 020040 020040 020040 000105 000000 000000 000000 010000      .       .E........
00030: 000000 000114 000000 000114 000036                                                   ...L...L..

LOG5962 RECORD 35 (%43, #23)
00000: 000005 000035 127163 007463 014003 040322 050523 046523 043503 040524 027120 052502 .....s.3..@.QSMSGCAT.PUB
00014: 020040 020040 020056 051531 051440 020040 020040 000105 000001 000000 000156 001400      .SYS     .E.....n..
00030: 000000 000020 000000 000001 000003                                                   ..........

LOG5962 RECORD 36 (%44, #24)
00000: 000005 000035 127163 007463 014003 040322 050523 047525 052040 020040 027040 020040 .....s.3..@.QSOUT   .
00014: 020040 020040 020056 020040 020040 020040 020040 000105 000400 000000 000000 010000      .       .E........
00030: 000000 000114 000000 000114 000036                                                   ...L...L..

LOG5962 RECORD 37 (%45, #25)
00000: 000005 000035 127163 007463 014003 040322 022123 052104 044516 020040 027040 020040 .....s.3..@.$STDIN  .
00014: 020040 020040 020056 020040 020040 020040 020040 001105 000000 000000 000000 010000      .       .E........
00030: 000000 000114 000000 000114 000036                                                   ...L...L..

LOG5962 RECORD 38 (%46, #26)
00000: 000005 000035 127163 007463 014003 040322 022123 052104 046111 051524 027040 020040 .....s.3..@.$STDLIST.
00014: 020040 020040 020056 020040 020040 020040 020040 001105 000000 000000 000000 010000      .       .E........
00030: 000000 000114 000000 000114 000036                                                   ...L...L..

LOG5962 RECORD 39 (%47, #27)
00000: 000005 000035 127163 007463 014004 040322 020040 020040 020040 020040 027114 040523 .....s.3..@.        .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 001000 000000 000000 000006 001400 VEGAS.TEST    ..........
00030: 000000 000000 000000 000000 000002                                                   ..........
```

```
LOG5962 RECORD 40 (%50, #28)
00000: 000004 000016 127163 007463 014006 040322 000020 000026 033553 000403 002135 000161 .....s.3..@.....7k...].q
00014: 000000 000012                                                                        ....

LOG5962 RECORD 41 (%51, #29)
00000: 000005 000035 127163 007463 020003 040322 046123 031440 020040 020040 027114 040523 .....s.3 .@.LS3     .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 000406 000400 000000 000002 001400 VEGAS.TEST    ..........
00030: 000000 000001 000000 000001 000004                                                   ..........

LOG5962 RECORD 42 (%52, #2A)
00000: 000005 000035 127163 007463 022001 040322 046120 020040 020040 020040 027114 040523 .....s.3$.@.LP      .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 001000 000000 000000 000040 020000 VEGAS.TEST    ....... .
00030: 000000 000004 000000 000001 000001                                                   ..........

LOG5962 RECORD 43 (%53, #2B)
00000: 000005 000035 127163 007463 022006 000000 052124 030470 020040 020040 027120 052502 .....s.3$...TT18    .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 020000 000001 000000 000002 001400      .SYS      .........
00030: 000000 000002 000000 000001 000001                                                   ..........

LOG5962 RECORD 44 (%54, #2C)
00000: 000010 000042 127163 007463 022007 000000 046507 051040 020040 020040 052105 051524 ...".s.3$...MGR     TEST
00014: 020040 020040 051503 047524 052040 020040 046120 020040 020040 020040 040322 102541     SCOTT   LP      @..a
00030: 020030 000015 000000 000006 000000 000040 007140 000000 000000 000000                 .......... .'......

LOG5962 RECORD 45 (%55, #2D)
00000: 000005 000035 127163 007463 022011 000000 020040 020040 020040 020040 027114 040523 .....s.3$...        .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 000145 002001 000000 000040 001400 VEGAS.TEST    .e..... ..
00030: 000000 000004 000000 000001 000001                                                   ..........

LOG5962 RECORD 46 (%56, #2E)
00000: 000005 000035 127163 007463 025001 040322 046123 032040 020040 020040 027114 040523 .....s.3*.@.LS4     .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 005400 000400 000000 000200 001400 VEGAS.TEST    ..........
00030: 000000 000000 000000 000000 000001                                                   ..........

LOG5962 RECORD 47 (%57, #2F)
00000: 000005 000035 127163 007463 026010 040322 045461 030465 030465 032071 027114 040523 .....s.3,.@.K1151549.LAS
00014: 053105 043501 051456 052105 051524 020040 020040 140000 002001 000000 001760 001400 VEGAS.TEST    ..........
00030: 000000 000024 000000 000003 000004                                                   ..........

LOG5962 RECORD 48 (%60, #30)
00000: 000005 000035 127163 007463 026010 040322 022123 052104 044516 020040 027040 020040 .....s.3,.@.$STDIN  .
00014: 020040 020040 020056 020040 020040 020040 020040 001006 000000 000000 000000 010000      .       ..........
00030: 000000 000130 000000 000130 000036                                                   ...X...X..

LOG5962 RECORD 49 (%61, #31)
00000: 000005 000035 127163 007463 026011 040322 022123 052104 046111 051524 027040 020040 .....s.3,.@.$STDLIST.
00014: 020040 020040 020056 020040 020040 020040 020040 001006 000000 000000 000000 010000      .       ..........
00030: 000000 000130 000000 000130 000036                                                   ...X...X..
```

```
LOG5962 RECORD 50 (%62, #32)
00000: 000005 000035 127163 007463 026011 040322 022123 052104 044516 054040 027040 020040 .....s.3,.@.$STDIWX .
00014: 020040 020040 020056 020040 020040 020040 001006 000000 000000 000000 010000 ..... .....
00030: 000000 000130 000000 000130 000036                                                 ...X...X..

LOG5962 RECORD 51 (%63, #33)
00000: 000005 000035 127163 007463 026011 040322 022123 052104 046111 051524 027040 020040 .....s.3,.@.$STDLIST.
00014: 020040 020040 020056 020040 020040 020040 001006 000000 000000 000000 010000 ..... .....
00030: 000000 000130 000000 000130 000036                                                 ...X...X..

LOG5962 RECORD 52 (%64, #34)
00000: 000004 000016 127163 007463 026401 040322 000033 000015 011501 000204 000425 000156 .....s.3-.@.....A.....n
00014: 000000 000003                                                                      ...

LOG5962 RECORD 53 (%65, #35)
00000: 000005 000035 127163 007463 031411 040322 046123 032440 020040 020040 027114 040523 .....s.33.@.LS5    .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 005400 000400 000000 000200 001400 VEGAS.TEST  .....
00030: 000000 000000 000000 000000 000002                                                 .....

LOG5962 RECORD 54 (%66, #36)
00000: 000005 000035 127163 007464 015001 000000 051524 047522 042440 020040 027120 052502 .....s.4....STORE  .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 000374 000001 000000 000414 001400    .SYS     .....
00030: 000000 000053 000000 000053 000001                                                 ...+...+..

LOG5962 RECORD 55 (%67, #37)
00000: 000005 000035 127163 007464 015003 040322 051524 047522 042440 020040 027120 052502 .....s.4..@.STORE  .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 001121 000001 000000 000414 001400    .SYS     .Q.....
00030: 000000 000017 000000 000017 000001                                                 .....

LOG5962 RECORD 56 (%70, #38)
00000: 000016 000037 127163 007464 015407 040322 000000 000001 003151 000401 000000 052101 .....s.4..@.....i....TA
00014: 050105 020040 020040 027040 020040 020040 020040 020040 020040 020040 020040 020040 PE      .
00030: 040501 040501 040501 040501 040501 000136                                          AAAAAAAAAAAA..

LOG5962 RECORD 57 (%71, #39)
00000: 000017 000041 127163 007464 015411 040322 000064 037461 032472 032462 027443 051462 ...!.s.4..@..4715:52/#S2
00014: 030460 027471 032057 046557 072556 072040 072141 070145 020157 063040 073157 066165 10/94/Mount tape of volu
00030: 066545 071545 072040 040501 040501 040501 020050 040516 051451                    meset AAAAAA (ANS)

LOG5962 RECORD 58 (%72, #3A)
00000: 000017 000041 127163 007465 006011 000000 000064 030465 035065 031457 030461 027522 ...!.s.5.....415:53/11/R
00014: 062545 066040 030454 020166 067554 020101 040501 040501 040440 024101 047123 024440 eel 1, vol AAAAAA (ANS)
00030: 066557 072556 072145 062040 067556 020114 042105 053043 020067                    mounted on LDEV# 7

LOG5962 RECORD 59 (%73, #3B)
00000: 000005 000035 127163 007465 013004 040322 043517 047504 020040 020040 027114 040523 .....s.5..@.GOOD    .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 000000 000000 000000 001400 000040 VEGAS.TEST  ..... ..
00030: 000000 000005 000000 000005 000003                                                 .....
```

APPENDIX C

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

```
LOG5962 RECORD 60 (%74, #3C)
00000: 000005 000035 127163 007465 013004 040322 051531 051514 044523 052040 027040 020040 .....s.5..@.SYSLIST .
00014: 020040 020040 020056 020040 020040 020040 020040 000266 000000 000000 000000 010000       .         ..........
00030: 000000 000156 000000 000156 000036                                                    ...n...n..

LOG5962 RECORD 61 (%75, #3D)
00000: 000005 000035 127163 007465 015405 040322 052101 050105 020040 020040 027040 020040 .....s.5..@.TAPE    .
00014: 020040 020040 020056 020040 020040 020040 020040 000000 000400 000000 000000 014000       .         ..........
00030: 000000 000002 000000 000001 000007                                                    ..........

LOG5962 RECORD 62 (%76, #3E)
00000: 000005 000035 127163 007465 015406 040322 020040 020040 020040 020040 027114 040523 .....s.5..@.        .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 020000 000000 000000 000021 001400 VEGAS.TEST     .........
00030: 000000 000000 000000 000000 000004                                                    ..........

LOG5962 RECORD 63 (%77, #3F)
00000: 000005 000035 127163 007465 015406 040322 022123 052104 044516 020040 027040 020040 .....s.5..@.$STDIN  .
00014: 020040 020040 020056 020040 020040 020040 020040 001266 000000 000000 000000 010000       .         ..........
00030: 000000 000156 000000 000156 000036                                                    ...n...n..

LOG5962 RECORD 64 (%100, #40)
00000: 000005 000035 127163 007465 015406 040322 022123 052104 046111 051524 027040 020040 .....s.5..@.$STDLIST.
00014: 020040 020040 020056 020040 020040 020040 020040 001266 000000 000000 000000 010000       .         ..........
00030: 000000 000156 000000 000156 000036                                                    ...n...n..

LOG5962 RECORD 65 (%101, #41)
00000: 000004 000016 127163 007465 015407 040322 000013 000000 030304 000104 000524 000136 .....s.5..@.....0..D.T..
00014: 000000 000002                                                                         ....

LOG5962 RECORD 66 (%102, #42)
00000: 000005 000035 127163 007466 010003 040322 046104 033040 020040 020040 027114 040523 .....s.6..@.LD6      .LAS
00014: 053105 043501 051456 052105 051524 020040 020040 005400 000400 000000 000200 001400 VEGAS.TEST     .........
00030: 000000 000000 000000 000000 000001                                                    ..........

LOG5962 RECORD 67 (%103, #43)
00000: 000017 000040 127163 007466 015007 040322 000061 030465 035065 032057 021523 031061 ... .s.6..@..115:54/#S21
00014: 030057 033467 027506 051117 046457 046507 051056 052105 051524 027473 046101 051440 0/77/FROM/MGR.TEST/;LAS
00030: 053105 043501 051440 042105 046517 020104 047516 042544                              VEGAS DEMO DONEd

LOG5962 RECORD 68 (%104, #44)
00000: 000005 000035 127163 007466 016005 040322 042120 030060 031103 020040 027104 040524 .....s.6..@.DP002C  .DAT
00014: 040504 050040 020056 052105 051524 020040 020040 010000 000001 000000 000137 001400 ADP  .TEST     ......._..
00030: 000000 000436 000000 000022 000004                                                    ..........

LOG5962 RECORD 69 (%105, #45)
00000: 000005 000035 127163 007466 016005 040322 042120 030060 031103 020040 027104 040524 .....s.6..@.DP002C  .DAT
00014: 040504 050040 020056 051531 051440 020040 020040 010000 000001 000000 000310 001400 ADP  .SYS      ..........
00030: 000000 001151 000000 000047 000002                                                    ...i...'..
```

```
LOG5962 RECORD 70 (%106, #46)
00000: 000004 000016 127163 007466 016005 040322 000000 000000 013260 000016 000307 000115 .....s.6..a.............M
00014: 000000 000005                                                                        ....

LOG5962 RECORD 71 (%107, #47)
00000: 000017 000030 127163 007466 016011 040322 000041 030465 035065 032057 021523 031061 ;;;.s.6..a..!15:54/#S21
00014: 030057 033467 027514 047507 047506 043040 047516 020114 042105 053040 021463 030001 0/77/LOGOFF ON LDEV #30.

LOG5962 RECORD 72 (%110, #48)
00000: 000003 000014 127163 007466 016011 040322 000230 000003 000000 000023 000000 000006 .....s.6..a.............

LOG5962 RECORD 73 (%111, #49)
00000: 000005 000035 127163 007466 016402 040322 022123 052104 046111 051524 027040 020040 .....s.6..a.$STDLIST.
00014: 020040 020040 020056 020040 020040 020040 020040 051521 000000 000000 000000 010000 .       SQ.........
00030: 000000 000166 000000 000166 000036                                     ...v...v..

LOG5962 RECORD 74 (%112, #4A)
00000: 000005 000035 127163 007466 016404 040322 022123 052104 044516 020040 027040 020040 .....s.6..a.$STDIN  .
00014: 020040 020040 020056 020040 020040 020040 020040 051521 000000 000000 000000 010000 .       SQ.........
00030: 000000 000166 000000 000166 000036                                     ...v...v..

LOG5962 RECORD 75 (%113, #4B)
00000: 000005 000035 127163 007466 016404 000000 052124 030460 020040 020040 027120 052502 .....s.6....TT10    .PUB
00014: 020040 020040 020056 051531 051440 020040 020040 020000 000001 000000 000002 001400     .SYS     ..........
00030: 000000 000002 000000 000001 000001                                     ..........
```

APPENDIX C

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

APPENDIX D

SAMPLE REPORTS FROM A SYSTEM LOGFILE ANALYSIS

The reports contained in Appendix D were extracted from a logfile analysis of a weekend batch run. The data was loaded into a data base from which the following reports were extracted. Comments relevant to each report are shown below:

Figure 1 -- Account Summary Report is derived from Job Summary Data

Figure 2 -- Device Summary Report is derived from FCLOSE data.

Figure 3 -- Job Summary Report sorted by CPU. Only jobs with over 250 CPU were selected. Note that J2223 is a big CPU user.

Figure 4 -- Job Summary Report sorted by FCLOSE count. Only jobs over 250 CPU were selected

Figure 5 -- Job Summary Report sorted by blocks of I-O. Only jobs over 250 CPU were selected. Note that job J2223 was a big user of I-O.

Figure 6 -- Detail File Close Report for J2223 with files over 100 blocks of I-O. Note that DBWR13 accounted for nearly 342,000 blocks of I-O.

Figure 7 -- Process Summary Report derived from process termination records with the program name being determined from fclose records. Note that QUERY.DPP.ELDEC is a big CPU user.

Figure 8 -- Process Report for the program QUERY.DPP.ELDEC only. Note that job J2223 used approximately 1900 CPU running QUERY.DPP.SYS

Figure 9 -- File Summary Report for files > 20,000 blocks I-O.

From the above, you can see that QUERY.DPP.ELDEC consumed approximately 1900 of 2900 CPU for the job J2223 and that approximately 342,000 of 366,000 blocks of I-O were done to a data base set DBWR13. With these facts as a starting point, the next step is to determine with the user if a more efficient way exists to do whatever is being done.

DATE: 04/26/87
TIME: 14:58:05

ACCOUNT SUMMARY REPORT
SORTED BY ACCOUNT

PAGE 1
REPORT-ID: RACCTS

ACCOUNT	DATE YYMMDD	********** TOTAL	BLOCKS OF DISC	I-O ********** TAPE	TERM	********** TOTAL	RECORDS OF DISC	I-O ********** TAPE	TERM	SEC CPU	MIN WALL	LINES SPOFL	PROC COUNT	FCLOSE COUNT
ELDEC	870417	560602	307406	7673	241051	1012503	688854	17290	241051	4304	1507	2659	580	6785
ELDEC	870418	4314054	3239061	61112	936568	8234352	6512784	85975	936568	28312	4986	4958	1431	22527
ELDEC	870419	189656	177355	6017	3162	350533	296440	11818	3162	2944	128	653	178	3195
GBSI	870417	11781	11631	100	0	49056	45981	500	0	223	50	75	125	1335
GBSI	870418	55928	55310	412	0	209657	196832	2060	0	918	206	309	515	6716
GBSI	870419	7030	6958	48	0	24922	23446	240	0	108	24	36	60	888
SYS	870417	1299	1235	45	0	12048	10869	163	0	52	254	47	41	365
SYS	870418	65106	17932	1577	43771	160940	86381	2787	43771	797	1394	369	216	2314
SYS	870419	676	654	16	0	4173	3781	52	0	17	4	16	14	168
TEST	870418	5766	4418	0	1348	16914	15566	0	1348	62	16	188	32	486
USERAG	870418	435501	395527	122	37792	582282	518204	2660	37792	3245	2205	6	41	707
USERCORP	870418	21947	4037	0	17910	25924	8014	0	17910	79	169	0	22	317

APPENDIX D -- Figure 1

DATE: 04/26/87 DEVICE SUMMARY REPORT PAGE 1
TIME: 14:58:51 SORTED BY LOGICAL DEVICE NUMBER REPORT-ID: RDEVS

LDEV LDEV	DEVICE TYPE	DEVICE NAME	TOTAL RECORDS	I-O BLOCKS	FCLOSE COUNT
1	3	DISC	1515593	736013	4743
2	3	DISC	1580636	542709	7113
3	3	DISC	2088250	682511	6979
4	3	DISC	1716096	1047181	12137
5	3	DISC	2869990	1035566	7103
7	24	TAPE	73954	73974	22
11	3	DISC	1612528	682784	10352
20	16	TERM	37216	37216	29
21	16	TERM	492203	492203	297
24	16	TERM	415290	415290	117
31	16	TERM	1818	1818	9
32	16	TERM	2163	2163	10
41	16	TERM	5940	5940	9
45	16	TERM	113050	113050	46
46	16	TERM	17115	17115	68
48	16	TERM	2992	2992	32
68	16	TERM	452	452	2
73	16	TERM	1966	1966	40
80	16	TERM	176292	176292	2
84	16	TERM	2851	2851	48
86	16	TERM	37495	37495	28
88	16	TERM	448	448	1
500	16	TERM	296611	296611	318
501	16	TERM	338704	338704	71
502	16	TERM	123708	123708	38
503	16	TERM	1898	1898	37
504	16	TERM	119578	119578	88
505	16	TERM	1774	1774	38
506	16	TERM	7530	7530	18
507	16	TERM	926	926	6

APPENDIX D -- Figure 2

DATE: 04/26/87
TIME: 15:00:42

JOB SUMMARY REPORT
SORTED BY SECONDS OF CPU

PAGE 1
REPORT-ID: RJOBSCPU

JOB#	ACCOUNT	GROUP	USER	JOBNAME	WHEN STARTED YYMMDD HH:MM	Q	LDEV IN	LDEV OUT	PRIORITY IN	PRIORITY OU	PRIORITY MAX	SEC CPU	MIN WALL	CR	FCLOSE	TOTAL I-O REC-DISC	TOTAL I-O BLK-DISC	TOTAL I-O BLK-LP	TOTAL I-O BLK-TERM
J2223	USERAG	CSE	MGR	ART	870418 12:02	D	10	27	8	0	187	2888	161	17	396	373702	366475	1637	0
J2354	ELDEC	DATAMCD	MGR	RP800MCJ	870418 18:30	C	10	27	15	6	152	2246	71	6	684	77048	73553	17256	0
J2250	ELDEC	DATAMCD	MGR	RP700MCJ	870418 13:23	C	10	27	9	6	152	2204	79	8	241	396090	161553	25	0
J2194	ELDEC	DATAMCD	MGR	RP100MCJ	870418 10:27	C	10	27	9	6	152	1480	70	20	175	171384	166725	57	0
J2337	ELDEC	DATAMCD	MGR	RP750MCJ	870418 17:48	C	10	27	15	6	152	1210	44	24	238	605748	530396	71	0
J1999	ELDEC	DATAIR	MGR	PA1301RJ	870417 21:16	E	10	27	15	6	152	1122	125	7	129	55827	39780	10	0
J2483	ELDEC	DATACEN	MGR	RP100CNJ	870418 23:29	D	10	27	9	6	187	911	27	14	129	81384	79220	35	0
J2121	ELDEC	DATACEN	MGR	SF090CEJ	870418 07:44	D	10	27	9	6	152	824	81	7	96	68914	67936	5867	0
J2216	ELDEC	DATAMCD	MGR	RP600MCJ	870418 11:42	D	10	27	15	6	187	756	27	4	71	63226	62366	14	0
J2493	ELDEC	DATACEN	MGR	RP600CNJ	870419 00:02	D	10	27	9	6	152	682	18	5	123	62786	61787	16	0
J2390	ELDEC	DATAMCD	MGR	EO034MCD	870418 19:48	D	10	27	9	5	187	637	71	4	90	78450	41882	17	0
J2332	ELDEC	DATAMCD	MGR	RP740MCJ	870418 17:37	D	10	27	9	6	187	614	74	4	111	11327	9445	399	0
J2158	ELDEC	DATAMCD	MGR	IC266MCJ	870418 09:04	D	10	27	9	6	187	536	60	4	65	133030	132095	5	0
J2416	ELDEC	DATAMCD	MGR	WH990MCJ	870418 20:59	D	10	27	9	6	187	506	71	4	114	1258	327	12	0
J2142	ELDEC	JOBMCD	MGR	WO201J	870418 08:22	D	10	27	9	0	187	505	82	5	44	74851	11763	4383	0
J2383	ELDEC	DATAMCD	MGR	RP790MCJ	870418 19:41	D	10	27	9	6	187	499	42	4	62	331151	179128	18	0
J2276	ELDEC	DATAMCD	MGR	RP721MCJ	870418 14:56	D	10	27	9	6	152	488	18	5	508	56378	48623	2518	0
J2524	ELDEC	DATACEN	MGR	RP740CNJ	870419 01:07	D	10	27	9	6	187	483	13	4	120	12496	11202	187	0
J2356	ELDEC	DATAMCD	MGR	SL100MCJ	870418 18:31	D	10	27	9	6	152	425	46	5	148	204930	58731	331	0
J2098	SYS	SYSOP	OP	PARTDUMP	870418 02:43	C	10	27	15	8	152	424	61	6	48	11935	5037	1637	0
J2423	ELDEC	DPX	MGR	SPOOKERJ	870418 21:07	C	10	27	1013	152		421	30	3	325	576022	69115	81	0
J2384	ELDEC	DATAMCD	MGR	RP780MCJ	870418 19:41	D	10	27	9	6	152	395	21	5	70	20093	7757	4	0
J2271	ELDEC	DATAMCD	MGR	RP720MCJ	870418 14:42	D	10	27	9	6	152	386	15	5	739	22186	20740	443	0
J1937	ELDEC	DATAMCD	MGR	OE500MCJ	870417 20:23	D	10	27	8	6	187	384	42	4	98	63530	60408	71	0
S1703	ELDEC	DATAMCD	MGR	BATCHB	870417 23:30	C	21	21	8	0	152	383	490	41	645	68940	8825	6	72813
J2050	ELDEC	DATACEN	MGR	SF550CEJ	870418 00:09	C	10	27	9	6	152	382	20	4	66	43144	42291	790	0
J2140	ELDEC	DATACEN	MGR	IC266CEJ	870418 08:21	D	10	27	9	6	187	382	56	4	72	113883	112952	5	0
J1986	ELDEC	DATAIR	MGR	PA020IRJ	870417 20:41	E	10	27	15	6	152	381	17	9	131	145937	82502	29	0
J2441	ELDEC	DATAMCD	MGR	EO033MCD	870418 21:53	D	10	27	9	5	187	379	25	4	87	84397	79693	3	0
J2166	ELDEC	DATACEN	MGR	SF590CEJ	870418 09:17	D	10	27	9	6	187	362	22	4	84	24134	23281	1267	0
J2111	ELDEC	DATAMCD	MGR	IC200MCJ	870418 07:28	D	10	27	9	6	152	325	28	7	168	13728	6412	13	0
J2517	ELDEC	DATACEN	MGR	RP730CNJ	870419 00:50	D	10	27	15	6	152	312	10	5	130	11481	9964	181	0
J2514	ELDEC	DATACEN	MGR	RP721CNJ	870419 00:39	D	10	27	15	6	152	305	11	5	433	20073	17167	870	0
S1759	ELDEC	DPP	MGR	EO034SSD	870418 19:50	C	502	502	8	0	152	285	244	1	38	996	149	0	123086
J2504	ELDEC	DATACEN	MGR	RP720CNJ	870419 00:27	D	10	27	15	6	152	276	13	5	303	13699	11840	406	0
J2397	ELDEC	DATAMCD	MGR	RP770MCJ	870418 20:03	D	10	27	9	6	187	274	21	4	81	38866	26634	23	0
J2112	ELDEC	DATACEN	MGR	IC200CNJ	870418 07:28	D	10	27	9	6	152	268	12	7	208	36974	33008	14	0
J2521	ELDEC	DATACEN	MGR	RP731CNJ	870419 00:59	D	10	27	15	6	152	263	9	6	133	16625	9506	386	0
J2327	ELDEC	DATAMCD	MGR	RP722MCJ	870418 17:26	C	10	27	9	6	152	260	11	5	134	13694	12577	36	0
J2058	ELDEC	DATAIR	MGR	EM800IRJ	870418 00:32	D	10	27	9	6	187	254	8	5	81	55878	54833	37	0
S1713	USERAG	CS	CS	TJ	870418 07:08	D	86	86	8	0	152	254	384	3	85	119259	19792	0	34237
												26371			7703	4375454	2817470		

APPENDIX D -- Figure 3

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

DATE: 04/26/87
TIME: 15:01:03

JOB SUMMARY REPORT
SORTED BY NUMBER OF FCLOSES

PAGE 1
REPORT-ID: RJOBSFCL

JOB#	ACCOUNT	GROUP	USER	JOBNAME	WHEN STARTED YYMMDD	HH:MM	Q	LDEV IN	LDEV OUT	PRIORITY IN	PRIORITY OU	PRIORITY MAX	SEC CPU	MIN WALL	CR	FCLOSE	TOTAL I-O REC-DISC	TOTAL I-O BLK-DISC	TOTAL I-O BLK-LP	TOTAL I-O BLK-TERM
J2271	ELDEC	DATAMCD	MGR	RP720MCJ	870418	14:42	D	10	27	9	6	152	386	15	5	739	22186	20740	443	0
J2354	ELDEC	DATAMCD	MGR	RP800MCJ	870418	18:30	C	10	27	15	6	152	2246	71	6	684	77048	73553	17256	0
S1703	ELDEC	DATAMCD	MGR	BATCHB	870417	23:30	C	21	21	8	0	152	383	490	41	645	68940	8825	6	72813
J2276	ELDEC	DATAMCD	MGR	RP721MCJ	870418	14:56	D	10	27	9	6	152	488	18	5	508	56378	48623	2518	0
J2514	ELDEC	DATACEN	MGR	RP721CNJ	870419	00:39	D	10	27	15	6	152	305	11	5	433	20073	17167	870	0
J2223	USERAG	CSE	MGR	ART	870418	12:02	D	10	27	8	0	187	2888	161	17	396	373702	366475	1637	0
J2423	ELDEC	DPX	MGR	SPOOKERJ	870418	21:07	C	10	27	1013		152	421	30	3	325	576022	69115	81	0
J2504	ELDEC	DATACEN	MGR	RP720CNJ	870419	00:27	D	10	27	15	6	152	276	13	5	303	13699	11840	406	0
J2250	ELDEC	DATAMCD	MGR	RP700MCJ	870418	13:23	C	10	27	9	6	152	2204	79	8	241	396090	161553	25	0
J2337	ELDEC	DATAMCD	MGR	RP750MCJ	870418	17:48	C	10	27	15	6	152	1210	44	24	238	605748	530396	71	0
J2112	ELDEC	DATACEN	MGR	IC200CNJ	870418	07:28	D	10	27	9	6	152	268	12	7	208	36974	33008	14	0
J2194	ELDEC	DATAMCD	MGR	RP100MCJ	870418	10:27	C	10	27	9	6	152	1480	70	20	175	171384	166725	57	0
J2111	ELDEC	DATAMCD	MGR	IC200MCJ	870418	07:28	D	10	27	9	6	152	325	28	7	168	13728	6412	13	0
J2356	ELDEC	DATAMCD	MGR	SL100MCJ	870418	18:31	D	10	27	9	6	152	425	46	5	148	204930	58731	331	0
J2327	ELDEC	DATAMCD	MGR	RP722MCJ	870418	17:26	C	10	27	9	6	152	260	11	5	134	13694	12577	36	0
J2521	ELDEC	DATACEN	MGR	RP731CNJ	870419	00:59	D	10	27	15	6	152	263	9	6	133	16625	9506	386	0
J1986	ELDEC	DATAIR	MGR	PA0201RJ	870417	20:41	E	10	27	15	6	152	381	17	9	131	145937	82502	29	0
J2517	ELDEC	DATACEN	MGR	RP730CNJ	870419	00:50	D	10	27	15	6	152	312	10	5	130	11481	9964	181	0
J1999	ELDEC	DATAIR	MGR	PA1301RJ	870417	21:16	E	10	27	15	6	152	1122	125	7	129	55827	39780	10	0
J2483	ELDEC	DATACEN	MGR	RP100CNJ	870418	23:29	D	10	27	9	6	187	911	27	14	129	81384	79220	35	0
J2493	ELDEC	DATACEN	MGR	RP600CNJ	870419	00:02	D	10	27	9	6	152	682	18	5	123	62786	61787	16	0
J2524	ELDEC	DATACEN	MGR	RP740CNJ	870419	01:07	D	10	27	9	6	187	483	13	4	120	12496	11202	187	0
J2416	ELDEC	DATAMCD	MGR	WH990MCJ	870418	20:59	D	10	27	9	6	187	506	71	4	114	1258	327	12	0
J2332	ELDEC	DATAMCD	MGR	RP740MCJ	870418	17:37	D	10	27	9	6	187	614	74	4	111	11327	9445	399	0
J1937	ELDEC	DATAMCD	MGR	OE500MCJ	870417	20:23	D	10	27	8	6	187	384	42	4	98	63530	60408	71	0
J2121	ELDEC	DATACEN	MGR	SF090CEJ	870418	07:44	D	10	27	9	6	152	824	81	7	96	68914	67936	5867	0
J2390	ELDEC	DATAMCD	MGR	E0034MCD	870418	19:48	D	10	27	9	5	187	637	71	4	90	78450	41882	17	0
J2441	ELDEC	DATAMCD	MGR	E0033MCD	870418	21:53	D	10	27	9	5	187	379	25	4	87	84397	79693	3	0
S1713	USERAG	CS	CS	TJ	870418	07:08	D	86	86	8	0	152	254	384	3	85	119259	19792	0	34237
J2166	ELDEC	DATACEN	MGR	SF590CEJ	870418	09:17	D	10	27	9	6	187	362	22	4	84	24134	23281	1267	0
J2397	ELDEC	DATAMCD	MGR	RP770MCJ	870418	20:03	D	10	27	9	6	187	274	21	4	81	38866	26634	23	0
J2058	ELDEC	DATAIR	MGR	EM8001RJ	870418	00:32	D	10	27	9	6	187	254	8	5	81	55878	54833	37	0
J2140	ELDEC	DATACEN	MGR	IC266CEJ	870418	08:21	D	10	27	9	6	187	382	56	4	72	113883	112952	5	0
J2216	ELDEC	DATAMCD	MGR	RP600MCJ	870418	11:42	D	10	27	15	6	187	756	27	4	71	63226	62366	14	0
J2384	ELDEC	DATAMCD	MGR	RP780MCJ	870418	19:41	D	10	27	9	6	152	395	21	5	70	20093	7757	4	0
J2050	ELDEC	DATACEN	MGR	SF550CEJ	870418	00:09	C	10	27	9	6	152	382	20	4	66	43144	42291	790	0
J2158	ELDEC	DATAMCD	MGR	IC266MCJ	870418	09:04	D	10	27	9	6	187	536	60	4	65	133030	132095	5	0
J2383	ELDEC	DATAMCD	MGR	RP790MCJ	870418	19:41	D	10	27	9	6	187	499	42	4	62	331151	179128	18	0
J2098	SYS	SYSOP	OP	PARTDUMP	870418	02:43	C	10	27	15	8	152	424	61	6	48	11935	5037	1637	0
J2142	ELDEC	JOBMCD	MGR	WO201J	870418	08:22	D	10	27	9	0	187	505	82	5	44	74851	11763	4383	0
S1759	ELDEC	DPP	MGR	E0034SSD	870418	19:50	C	502	502	8	0	152	285	244	1	38	996	149	0	123086
TOTALS													26371			7703	4375454	2817470		

APPENDIX D -- Figure 4

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

DATE: 04/26/87
TIME: 15:01:21

JOB SUMMARY REPORT
SORTED BY TOTAL BLOCKS DISC I-O

PAGE 1
REPORT-ID: RJOBS

JOB#	ACCOUNT	GROUP	USER	JOBNAME	WHEN STARTED YYMMDD	HH:MM	Q	LDEV IN	LDEV OUT	PRIORITY IN	PRIORITY OU	PRIORITY MAX	SEC CPU	MIN WALL	CR	FCLOSE	TOTAL I-O REC-DISC	TOTAL I-O BLK-DISC	TOTAL I-O BLK-LP	TOTAL I-O BLK-TERM
J2337	ELDEC	DATAMCD	MGR	RP750MCJ	870418	17:48	C	10	27	15	6	152	1210	44	24	238	605748	530396	71	0
J2223	USERAG	CSE	MGR	ART	870418	12:02	D	10	27	8	0	187	2888	161	17	396	373702	366475	1637	0
J2383	ELDEC	DATAMCD	MGR	RP790MCJ	870418	19:41	D	10	27	9	6	187	499	42	4	62	331151	179128	18	0
J2194	ELDEC	DATAMCD	MGR	RP100MCJ	870418	10:27	C	10	27	9	6	152	1480	70	20	175	171384	166725	57	0
J2250	ELDEC	DATAMCD	MGR	RP700MCJ	870418	13:23	C	10	27	9	6	152	2204	79	8	241	396090	161553	25	0
J2158	ELDEC	DATAMCD	MGR	IC266MCJ	870418	09:04	D	10	27	9	6	187	536	60	4	65	133030	132095	5	0
J2140	ELDEC	DATACEN	MGR	IC266CEJ	870418	08:21	D	10	27	9	6	187	382	56	4	72	113883	112952	5	0
J1986	ELDEC	DATAIR	MGR	PA0201RJ	870417	20:41	E	10	27	15	6	152	381	17	9	131	145937	82502	29	0
J2441	ELDEC	DATAMCD	MGR	E0033MCD	870418	21:53	D	10	27	9	5	187	379	25	4	87	84397	79693	3	0
J2483	ELDEC	DATACEN	MGR	RP100CNJ	870418	23:29	D	10	27	9	6	187	911	27	14	129	81384	79220	35	0
J2354	ELDEC	DATAMCD	MGR	RP800MCJ	870418	18:30	C	10	27	15	6	152	2246	71	6	684	77048	73553	17256	0
J2423	ELDEC	DPX	MGR	SPOOKERJ	870418	21:07	C	10	27	1013		152	421	30	3	325	576022	69115	81	0
J2121	ELDEC	DATACEN	MGR	SF090CEJ	870418	07:44	D	10	27	9	6	152	824	81	7	96	68914	67936	5867	0
J2216	ELDEC	DATAMCD	MGR	RP600MCJ	870418	11:42	D	10	27	15	6	187	756	27	4	71	63226	62366	14	0
J2493	ELDEC	DATACEN	MGR	RP600CNJ	870419	00:02	D	10	27	9	6	152	682	18	5	123	62786	61787	16	0
J1937	ELDEC	DATAMCD	MGR	OE500MCJ	870417	20:23	D	10	27	8	6	187	384	42	4	98	63530	60408	71	0
J2356	ELDEC	DATAMCD	MGR	SL100MCJ	870418	18:31	D	10	27	9	6	152	425	46	5	148	204930	58731	331	0
J2058	ELDEC	DATAIR	MGR	EM8001RJ	870418	00:32	D	10	27	9	6	187	254	8	5	81	55878	54833	37	0
J2276	ELDEC	DATAMCD	MGR	RP721MCJ	870418	14:56	D	10	27	9	6	152	488	18	5	508	56378	48623	2518	0
J2050	ELDEC	DATACEN	MGR	SF550CEJ	870418	00:09	C	10	27	9	6	152	382	20	4	66	43144	42291	790	0
J2390	ELDEC	DATAMCD	MGR	E0034MCD	870418	19:48	D	10	27	9	5	187	637	71	4	90	78450	41882	17	0
J1999	ELDEC	DATAIR	MGR	PA1301RJ	870417	21:16	E	10	27	15	6	152	1122	125	7	129	55827	39780	10	0
J2112	ELDEC	DATACEN	MGR	IC200CNJ	870418	07:28	D	10	27	9	6	152	268	12	7	208	36974	33008	14	0
J2397	ELDEC	DATAMCD	MGR	RP770MCJ	870418	20:03	D	10	27	9	6	187	274	21	4	81	38866	26634	23	0
J2166	ELDEC	DATACEN	MGR	SF590CEJ	870418	09:17	D	10	27	9	6	187	362	22	4	84	24134	23281	1267	0
J2271	ELDEC	DATAMCD	MGR	RP720MCJ	870418	14:42	D	10	27	9	6	152	386	15	5	739	22186	20740	443	0
S1713	USERAG	CS	CS	TJ	870418	07:08	D	86	86	8	0	152	254	384	3	85	119259	19792	0	34237
J2514	ELDEC	DATACEN	MGR	RP721CNJ	870419	00:39	D	10	27	15	6	152	305	11	5	433	20073	17167	870	0
J2327	ELDEC	DATAMCD	MGR	RP722MCJ	870418	17:26	C	10	27	9	6	152	260	11	5	134	13694	12577	36	0
J2504	ELDEC	DATACEN	MGR	RP720CNJ	870419	00:27	D	10	27	15	6	152	276	13	5	303	13699	11840	406	0
J2142	ELDEC	JOBMCD	MGR	WO201J	870418	08:22	D	10	27	9	0	187	505	82	5	44	74851	11763	4383	0
J2524	ELDEC	DATACEN	MGR	RP740CNJ	870419	01:07	D	10	27	9	6	187	483	13	4	120	12496	11202	187	0
J2517	ELDEC	DATACEN	MGR	RP730CNJ	870419	00:50	D	10	27	15	6	152	312	10	5	130	11481	9964	181	0
J2521	ELDEC	DATACEN	MGR	RP731CNJ	870419	00:59	D	10	27	15	6	152	263	9	6	133	16625	9506	386	0
J2332	ELDEC	DATAMCD	MGR	RP740MCJ	870418	17:37	D	10	27	9	6	187	614	74	4	111	11327	9445	399	0
S1703	ELDEC	DATAMCD	MGR	BATCH8	870417	23:30	C	21	21	8	0	152	383	490	41	645	68940	8825	6	72813
J2384	ELDEC	DATAMCD	MGR	RP780MCJ	870418	19:41	D	10	27	9	6	152	395	21	5	70	20093	7757	4	0
J2111	ELDEC	DATAMCD	MGR	IC200MCJ	870418	07:28	D	10	27	9	6	152	325	28	7	168	13728	6412	13	0
J2098	SYS	SYSOP	OP	PARTDUMP	870418	02:43	C	10	27	15	8	152	424	61	6	48	11935	5037	1637	0
J2416	ELDEC	DATAMCD	MGR	WH990MCJ	870418	20:59	D	10	27	9	6	187	506	71	4	114	1258	327	12	0
S1759	ELDEC	DPP	MGR	E0034SSD	870418	19:50	C	502	502	8	0	152	285	244	1	38	996	149	0	123086

APPENDIX D -- Figure 5

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

DATE: 04/26/87
TIME: 15:04:18

TYPE-05 FILE CLOSE REPORT
SORTED BY ACCOUNT, GROUP, FILE

WHEN CLOSED YYMMDD	HH:MM:SS.T	JOB#	JOBNAME	ACCOUNT	GROUP	FILENAME	******** FILE DISPO	DOMAIN ********	SECTORS	DEV TYPE	*** FILE I-O *** RECORDS	BLOCKS	DEV NUM
870418	12:14:10.8	J2223		USERAG	.CSE	.CS001A0	0	0	880	32	1659	213	1
870418	12:36:04.9	J2223		USERAG	.CSE	.CS005A0	0	0	840	32	1526	203	5
870418	12:45:19.5	J2223		USERAG	.CSE	.CS009A0	0	0	832	32	1519	201	4
870418	12:55:45.9	J2223		USERAG	.CSE	.CS012A0	0	0	1244	32	2724	304	1
870418	12:22:13.6	J2223		USERAG	.CSE	.CS04BA0	0	0	804	32	1703	194	1
870418	13:31:20.0	J2223		USERAG	.CSE	.DBWR01	0	1	11025	3	7392	7392	5
870418	13:48:16.4	J2223		USERAG	.CSE	.DBWR01	0	1	11025	3	2070	2070	5
870418	13:31:20.3	J2223		USERAG	.CSE	.DBWR02	0	1	610	3	234	234	11
870418	13:31:20.5	J2223		USERAG	.CSE	.DBWR03	0	1	215	3	227	227	1
870418	13:31:20.6	J2223		USERAG	.CSE	.DBWR04	0	1	20	3	230	230	5
870418	13:31:20.6	J2223		USERAG	.CSE	.DBWR05	0	1	6	3	230	230	3
870418	13:31:20.7	J2223		USERAG	.CSE	.DBWR06	0	1	50	3	1470	1470	4
870418	13:31:20.7	J2223		USERAG	.CSE	.DBWR07	0	1	80	3	235	235	5
870418	13:31:20.8	J2223		USERAG	.CSE	.DBWR09	0	1	16	3	271	271	1
870418	13:31:20.8	J2223		USERAG	.CSE	.DBWR11	0	1	410	3	258	258	3
870418	13:31:20.8	J2223		USERAG	.CSE	.DBWR12	0	1	410	3	230	230	4
870418	13:31:20.8	J2223		USERAG	.CSE	.DBWR13	0	1	104392	3	230922	230922	5
870418	13:48:16.5	J2223		USERAG	.CSE	.DBWR13	0	1	104392	3	24217	24217	5
870418	14:01:29.3	J2223		USERAG	.CSE	.DBWR13	0	1	104392	3	22122	22122	5
870418	14:12:43.3	J2223		USERAG	.CSE	.DBWR13	0	1	104392	3	22122	22122	5
870418	14:24:14.9	J2223		USERAG	.CSE	.DBWR13	0	1	104392	3	22122	22122	5
870418	14:42:12.8	J2223		USERAG	.CSE	.DBWR13	0	1	104392	3	22122	22122	5
870418	13:31:20.8	J2223		USERAG	.CSE	.DBWR14	0	1	11140	3	334	334	11
870418	13:31:20.9	J2223		USERAG	.CSE	.DBWR16	0	1	6952	3	3479	3479	4
870418	12:14:09.9	J2223		USERAG	.CSE	.QSSELECT	0	0	388	3	144	144	3
870418	12:22:12.9	J2223		USERAG	.CSE	.QSSELECT	0	0	388	3	144	144	4
870418	12:36:04.5	J2223		USERAG	.CSE	.QSSELECT	0	0	388	3	144	144	2
870418	12:45:18.1	J2223		USERAG	.CSE	.QSSELECT	0	0	388	3	144	144	2
870418	12:55:45.4	J2223		USERAG	.CSE	.QSSELECT	0	0	388	3	144	144	4
870418	12:13:25.6	J2223		USERAG	.CSE	.SORTSCR	0	0	244	3	428	428	5
870418	12:21:52.4	J2223		USERAG	.CSE	.SORTSCR	0	0	241	3	356	356	3
870418	12:29:01.6	J2223		USERAG	.CSE	.SORTSCR	0	0	83	3	154	154	11
870418	12:35:32.0	J2223		USERAG	.CSE	.SORTSCR	0	0	244	3	480	480	4
870418	12:44:13.6	J2223		USERAG	.CSE	.SORTSCR	0	0	157	3	210	210	5
870418	12:54:41.7	J2223		USERAG	.CSE	.SORTSCR	0	0	244	3	464	464	3
870418	13:06:15.8	J2223		USERAG	.CSE	.SORTSCR	0	0	83	3	130	130	3
870418	13:48:12.1	J2223		USERAG	.CSE	.SORTSCR	0	0	283	3	130	130	5
870418	14:12:39.4	J2223		USERAG	.CSE	.SORTSCR	0	0	369	3	144	144	5
											372634	364618	

APPENDIX D -- Figure 6

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

DATE: 04/26/87 PROCESS SUMMARY REPORT PAGE 1
TIME: 14:58:51 SORTED BY SECONDS OF CPU REPORT-ID: RPROCSUM

NAME	GROUP	ACCT	SEC-CPU	COUNT
RP700P	.DPP	.ELDEC	2431	2
QUERY	.DPP	.ELDEC	2405	29
RP100P	.DPP	.ELDEC	2369	2
RP800P	.DPP	.ELDEC	2282	2
RP600P	.DPP	.ELDEC	1427	2
DP025P	.DPP	.ELDEC	1403	29
RP750P	.DPP	.ELDEC	1198	2
PA130P	.DPP	.ELDEC	1115	1
SE010P	.DPP	.ELDEC	1115	743
RP740P	.DPP	.ELDEC	1086	2
IC266P	.DPP	.ELDEC	907	2
SPOOK5	.PUB	.SYS	853	39
SF090P	.DPP	.ELDEC	818	1
RP721P	.DPP	.ELDEC	780	2
EO034P	.DPP	.ELDEC	759	4
RP720P	.DPP	.ELDEC	650	2
IC200P	.DPP	.ELDEC	576	2
STORE	.PUB	.SYS	549	15
WH990P	.DPP	.ELDEC	516	2
RP790P	.DPP	.ELDEC	505	2
WO200P	.DPP	.ELDEC	500	1
STDTRN1P	.STDTRAN	.GBSI	470	140
SL100P	.DPP	.ELDEC	422	2
SF105P	.DPP	.ELDEC	413	2
DP012P	.DPP	.ELDEC	410	378
SF590P	.DPP	.ELDEC	398	2
RP780P	.DPP	.ELDEC	391	2
EO033P	.DPP	.ELDEC	386	6
OE500P	.DPP	.ELDEC	380	1
SF550P	.DPP	.ELDEC	378	1
PA020P	.DPP	.ELDEC	372	1
DP100P	.DPP	.ELDEC	352	262
SC320P	.DPP	.ELDEC	325	2
IC300P	.DPP	.ELDEC	313	2
PO110P	.DPP	.ELDEC	308	2
RP730P	.DPP	.ELDEC	305	1
IC350P	.DPP	.ELDEC	303	2
PVINIT	.PUB	.SYS	282	1
PVSPROG	.PUB	.USERAG	276	5
DP016P	.DPP	.ELDEC	272	199
RP770P	.DPP	.ELDEC	270	2
SF710P	.DPP	.ELDEC	270	2
RP722P	.DPP	.ELDEC	257	5
RP731P	.DPP	.ELDEC	256	1

APPENDIX D -- Figure 7

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

DATE: 04/26/87
TIME: 15:09:55

PROCESS SUMMARY REPORT
SORTED BY ACCOUNT, GROUP, PROGRAM

PAGE 1
REPORT-ID: RPROCS

PROGRAM	GROUP	ACCOUNT	WHEN STARTED YYMMDD HH:MM	JOB#	JOBNAME	PIN	SEC CPU	SEGMENTS SL	SEGMENTS PROC	MAX STACK	MAX DSEG	VIRTUAL STORAGE
QUERY	.DPP	.ELDEC	870418 13:31	J2223		27	537	38	16	30058	259	1179
QUERY	.DPP	.ELDEC	870418 13:12	S1670		160	331	38	16	14910	259	2794
QUERY	.DPP	.ELDEC	870418 13:06	J2223		162	224	38	16	30058	259	1179
QUERY	.DPP	.ELDEC	870418 12:36	J2223		77	214	38	16	30058	259	1179
QUERY	.DPP	.ELDEC	870418 12:14	J2223		123	213	38	16	30058	259	1179
QUERY	.DPP	.ELDEC	870418 12:55	J2223		137	212	38	16	30058	259	1179
QUERY	.DPP	.ELDEC	870418 12:45	J2223		40	209	38	16	30058	259	1179
QUERY	.DPP	.ELDEC	870418 12:22	J2223		73	205	38	16	30058	259	1179
QUERY	.DPP	.ELDEC	870418 12:29	J2223		67	167	38	16	30058	259	1179
QUERY	.DPP	.ELDEC	870418 13:22	J2248		60	31	38	16	14222	259	922
QUERY	.DPP	.ELDEC	870418 13:32	J2223		120	16	38	16	30058	259	1179
QUERY	.DPP	.ELDEC	870419 00:21	J2499		112	8	38	16	14222	259	1440
QUERY	.DPP	.ELDEC	870418 16:51	S1743		186	4	38	16	14762	259	2357
QUERY	.DPP	.ELDEC	870418 23:44	S1765		127	4	38	16	14762	259	2098
QUERY	.DPP	.ELDEC	870419 01:22	S1743		88	3	38	16	14762	259	2098
QUERY	.DPP	.ELDEC	870417 20:46	S1662		118	2	38	16	14762	91	377
QUERY	.DPP	.ELDEC	870417 22:05	S1670		33	2	38	16	14762	91	377
QUERY	.DPP	.ELDEC	870417 23:02	S1662		32	2	38	16	14762	91	377
QUERY	.DPP	.ELDEC	870418 01:17	S1703		121	2	38	16	14094	91	377
QUERY	.DPP	.ELDEC	870418 07:56	S1716		141	2	38	16	14762	259	870
QUERY	.DPP	.ELDEC	870418 15:31	S1670		167	2	38	16	14762	259	1154
QUERY	.DPP	.ELDEC	870418 15:48	S1670		114	2	38	16	14790	259	1154
QUERY	.DPP	.ELDEC	870418 16:30	S1743		150	2	38	16	14762	259	1154
QUERY	.DPP	.ELDEC	870418 17:47	S1743		180	2	38	16	14222	259	2066
QUERY	.DPP	.ELDEC	870418 21:09	S1743		118	2	38	16	14762	91	377
QUERY	.DPP	.ELDEC	870418 23:18	S1743		185	2	38	16	14762	259	895
QUERY	.DPP	.ELDEC	870418 23:59	S1743		100	2	38	16	14762	259	895
QUERY	.DPP	.ELDEC	870419 00:00	S1743		40	2	38	16	14094	259	895
QUERY	.DPP	.ELDEC	870418 02:17	J2088		134	1	38	16	13198	0	256
							2405					

APPENDIX D -- Figure 8

SYSTEM LOGFILES: WHAT THEY CAN TELL YOU

APPENDIX D -- Figure 9

DATE: 04/28/87 FILE SUMMARY REPORT PAGE 1
TIME: 16:54:30 SORTED BY IO (BLOCKS) REPORT-ID: RFILESIO

ACCOUNT	GROUP	FILENAME	LDEV	DEVICE TYPE	DEVICE NAME	*** TOTAL I-O *** RECORDS	BLOCKS	FCLOSE COUNT
ELDEC	DATAMCD	SORTSCR	5	3	DISC	963088	963088	113
ELDEC	DATAIR	SORTSCR	2	3	DISC	351398	351398	14
USERAG	CSE	DBWR13	5	3	DISC	348156	348156	8
ELDEC	DATACEN	SORTSCR	5	3	DISC	173470	173470	103
ELDEC	DATAMCD	DBWIP05	4	3	DISC	145822	145822	15
ELDEC	DATAMCD	DBENG26	2	3	DISC	132523	132523	28
ELDEC	DATAMCD	DBMRPW01	1	3	DISC	114242	114242	17
ELDEC	DATADP		5	3	DISC	803234	102818	289
ELDEC	DPP		3	3	DISC	758923	92698	422
ELDEC	DATACEN	DBENG26	3	3	DISC	90681	90681	24
ELDEC	DATAMCD	DBENG01	11	3	DISC	89089	89089	19
ELDEC	DATAMCD		4	3	DISC	677109	77891	990
ELDEC	DATACEN	DBWIP05	5	3	DISC	74435	74435	8
ELDEC	DATACEN	DBMRPW01	11	3	DISC	68641	68641	15
ELDEC	DATAMCD	DBMRP01	1	3	DISC	63635	63635	6
ELDEC	DATACEN		1	3	DISC	456520	59541	651
ELDEC	DATACEN	DBWIP01	5	3	DISC	56266	56266	20
ELDEC	DATAMCD	DBMRP02	3	3	DISC	43937	43937	9
GBSI	STDTRAN	STDTRAND	1	3	DISC	41860	41860	140
ELDEC	DATAMCD	DBWIP07	11	3	DISC	41270	41270	11
ELDEC	DATADP	DBCPU30	3	3	DISC	38550	38550	1044
SYS	DATADP	DP002C	5	3	DISC	555863	35373	749
ELDEC	DATAMCD	DBMRPW08	1	3	DISC	34970	34970	8
ELDEC	DATAIR	DBEMX04	3	3	DISC	32839	32839	9
ELDEC	PUB	SL	2	3	DISC	31179	31179	185
ELDEC	DATACEN	DBWIP31	11	3	DISC	30134	30134	20
ELDEC	DATADP	DBCPU	11	3	DISC	29855	29855	2054
ELDEC	DATACEN	DBWIP30	1	3	DISC	27562	27562	23
ELDEC	DATACEN	DBENG04	5	3	DISC	25190	25190	5
ELDEC	JOBMCD		4	3	DISC	204923	24536	20
ELDEC	DATADP	DBTBL02	4	3	DISC	23709	23709	345
ELDEC	DATAMCD	DBOR05	2	3	DISC	23589	23589	5
ELDEC	DATAMCD	DBWIP30	1	3	DISC	23500	23500	21
ELDEC	DATAMCD	DBPO04	2	3	DISC	22619	22619	3
ELDEC	DATAMCD	DBHIST26	4	3	DISC	22415	22415	14
ELDEC	DATAMCD	DBWIP42	5	3	DISC	22212	22212	20
ELDEC	DATACEN	DBOR05	3	3	DISC	21644	21644	2
ELDEC	DATACEN	DBPO04	11	3	DISC	21393	21393	1
ELDEC	DATACEN	DBENG01	2	3	DISC	20830	20830	13
ELDEC	DATAMCD	DBMRPW07	11	3	DISC	20455	20455	6
						6727730	3664015	7449

USING SUBPROGRAMS TO IMPROVE PERFORMANCE

George B. Scott
ELDEC CORPORATION
16700 13th Avenue West
Lynnwood, WA 98046-0100

I. INTRODUCTION

The question "Why use subprograms?" quickly leads us to the question of "Why build a large program?". This paper will present some advantages that a large program structure can provide over many small programs and why the use of subprograms to contain application code has provided a mechanism to facilitate management of change control and provide excellent performance characteristics.

The remainder of this paper is divided into the following sections.

Section	Title
II	Application Scope / System Overview
III	Design Requirements, Considerations and Guidelines
IV	The Process Tree
V	Functions of the On-Line Application Program
VI	Structure of the On-Line Application Program
VII	Changing/Expanding the Application Program
VIII	Performance vs. Design
IX	Results: The Good, The Bad and The Ugly
X	Summary

II. APPLICATION SCOPE / SYSTEM OVERVIEW

Applications and Subsystems

ELDEC Corporation is an aerospace manufacturing/engineering company which designs and manufactures sensing systems, power conversion units, and monitoring and control devices. The corporate business, manufacturing and accounting applications are installed on a network of HP3000 Series 70 computers.

Figure 1 shows the basic subsystems and application areas included within the large program which has been built to service on-line functions for all of these application areas. The program, named "BAP" for Business Application Processor, continues to grow in scope as the system is expanded to eliminate manual or stand alone systems.

Accounts Payable	Purchasing
Accounts Receivable	Quality Assurance
Air Traffic Association	Receiving
Bill-of-Material	Sales Analysis
Document Inventory	Sales Order Management
Engineering Change Orders	Security (DP)
Hazardous Material (MSDS)	Shipping
Inventory Control	Shop Floor Dispatch
Marketing	Spares Pricing
Material Requirements (MRP)	Standard Cost
Master Scheduling	User Documentation (DP)
Order Entry	Vendor Approval
Payroll	Warranty & Repair
Personnel	Work-in-Process
Process Routing	Work Order Release
Provisioning	Wirelists

Figure 1 -- Current Application Scope

System Overview: Facts and Figures

To give you a better understanding of the system, let me describe some other aspects. The company is divided into three divisions. Each division uses a Series 70 for processing data. Central engineering support, manufacturing support, corporate accounting and industrial relations all share one of the divisions computers. Each manufacturing division has a set of data bases. Corporate data bases for customer, vendor, security (DP), marketing and a subset of employee information are replicated on each system. The data in Figure 2 will give you a better picture of the size of the combined three systems.

Type	Size/Quantity	Description
Data:	8,565 Mb	Disc Storage
	542	Unique Data Base Sets
	4,612	Data Base Fields
	2,014	Unique Data Items
Hardware:	3	CPU's -- Series 70
	1	CPU -- Series 42 (for DP)
	1	X.25 Switch linked to each CPU
	350	Terminals/PC's
	4	System Printers
	31	Small Printers (Laserjets, etc.)
Software:	821	Stand Alone Programs
	729	Standard Reports
	1,015	On-Line Commands
	666	V/3000 Forms
	1,648	Source Code Files
	1,541,800	Lines of Source Code
BAP	71	Segments,Program
	151	" ,Group SL
	13	" ,Pub SL
	235 *	Total Segments
	813,000	Object Code,Program (Bytes)
	4,183,000	" ,Group SL (Bytes)
	83,000	" ,Pub SL (Bytes)
	5,079,000 *	Total Size (Bytes)
	621	Total Application Subprograms

Figure 2 -- System Overview: Facts & Figures

III. DESIGN REQUIREMENTS, CONSIDERATIONS AND GUIDELINES

To help you better understand why the current design was selected from several options, let me describe some of the requirements which contributed to fundamental design decisions.

1. No user MPE knowledge was to be required.
2. Command driven, not menu driven
3. On-line inquiry and update.
4. Audit/Security provisions.
5. Recovery capability for logical transactions.
6. Expandable and easily modifiable.
7. Five second or less median transaction time.

Requirement 1 -- No User MPE Knowledge Was to Be Required

The first requirement was that users would not be required to learn any aspect of MPE, e.g. how to sign-on, build files, use sort, run programs, know JCL, or any of the other fun things we system people get to learn. Users would be trained on how to use the application commands (screens) associated with the functional aspects of their job.

In essence, this meant that a technique (process), had to be created to interface with the user. From this process, all other requirements had to be satisfied. The first decision was whether one process could handle all terminals or should there be a process associated with each user terminal. Since it was anticipated that hundreds of terminals would eventually be used, (originally, only 36 were used), it was decided that the one terminal/one process approach seemed less likely to lead to a bottleneck.

Requirement 2 -- Command Driven, Not Menu Driven

The second requirement was that the user would be able to execute any command for which he/she was authorized without going through a menu or screen of commands. The real implications behind this requirement are discussed in requirement 7, a five second or less response time.

Requirement 3 -- On-Line Inquiry and Update

Simply stated, the third requirement meant that updates had to be performed when the user entered the command/data. We couldn't cheat and do them in a batch mode at night. This requirement played a very little role in our design; however, it had a very significant impact on requirement 7 for a five second or less response time.

Requirement 4 -- Provisions for Security/Audit

The fourth requirement was that each user would be authorized at a command level within an organization (division). For example, a user might be able to add an inventory receipt in one division, inquire on inventory on-hand in another division, and be excluded from inventory information in a third division. Thus security had to be checked at a division/employee/command level.

The second phase of this requirement was that all transactions had to be logged to provide an audit trail of who did what, when.

Requirement 5 -- Recovery of Logical Transactions

The fifth requirement was to provide a mechanism so that the system could be recovered from a known point by reprocessing the transactions which had occurred. The important part of this is the concept of logical recovery. For example, if application code containing an error was installed and caused the data base to be updated incorrectly, it would be useless to run some version of IMAGE recovery which would only duplicate the error.

The goal was to correct the application code and re-install it, restore the data base to a known good point, and reprocess all transactions (with the associated edits) to produce a data base which was correctly updated.

The primary consideration which resulted from this requirement was that each transaction had to stand alone. It had to have all the data necessary to successfully complete the transaction. Data could not be spread among several transactions.

Requirement 6 -- Expandable and Easily Modifiable

The sixth requirement, for an application system that could easily be expanded or modified, meant we had to derive a design with at least a 200% expansion capability. Thus far the size of the On-Line program has expanded to approximately 500% of its original size and is growing at the rate of approximately 125% (of the original) per year.

The second part of this requirement was the system had to accommodate software changes with a minimum of time and effort. Installation of new commands and application code should be able to be done routinely on a daily basis and in an emergency should be able to be done in less than an hour. Although we currently do more than is required to install changes, you will see in Section VII that the use of subprograms with a driver is a very elegant way to introduce code additions and modifications expeditiously.

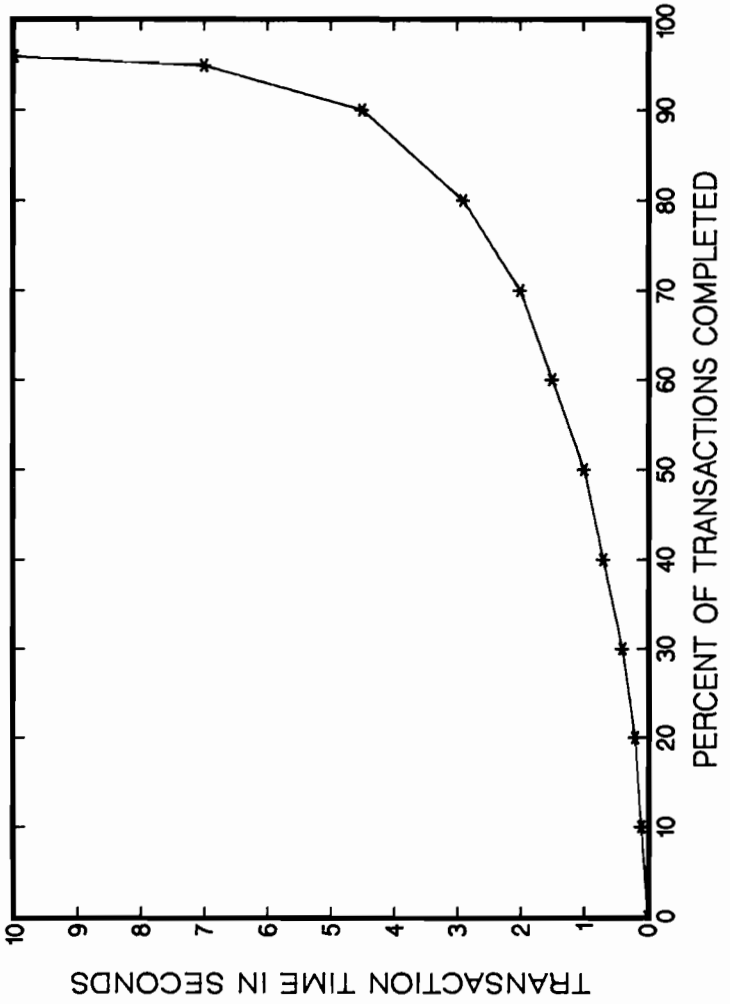


Figure 3 -- Response Time vs. Percent Transactions Processed

Requirement 7 -- Five Second or Less Median Response Time

Today, our median response time is 1 second. See Figure 3 for a graph of transaction time vs. percent of transactions completed.

The requirement for a five second or less response time played the greatest single role in our design considerations. Several decisions were made based on the anticipation of significant growth in the application software. At the time the system was designed, we had a single Series 44. Following are some design decisions and guidelines that affected our design; otherwise known as the things on which we bet our behinds.

1. I-O is costly in time and CPU. Use memory when high activity and a low volume of data is involved.

2. The cost of memory will continue to drop. The size of available memory will continue to expand.

3. Process creation is a very expensive operation in CPU and time. Minimize creations/terminations.

4. Minimize data base opens and closes, which are costly in CPU and time.

5. Minimize inter-process mail/communication for simplicity.

6. Provide global subroutines (subprograms) for as many highly repetitive routines as possible.

7. Application code subprograms should not execute program setup or initialization logic.

8. Any application command should be executable from some single point of origin in a rapid fashion.

9. KISS, or "Keep it simple, stupid".

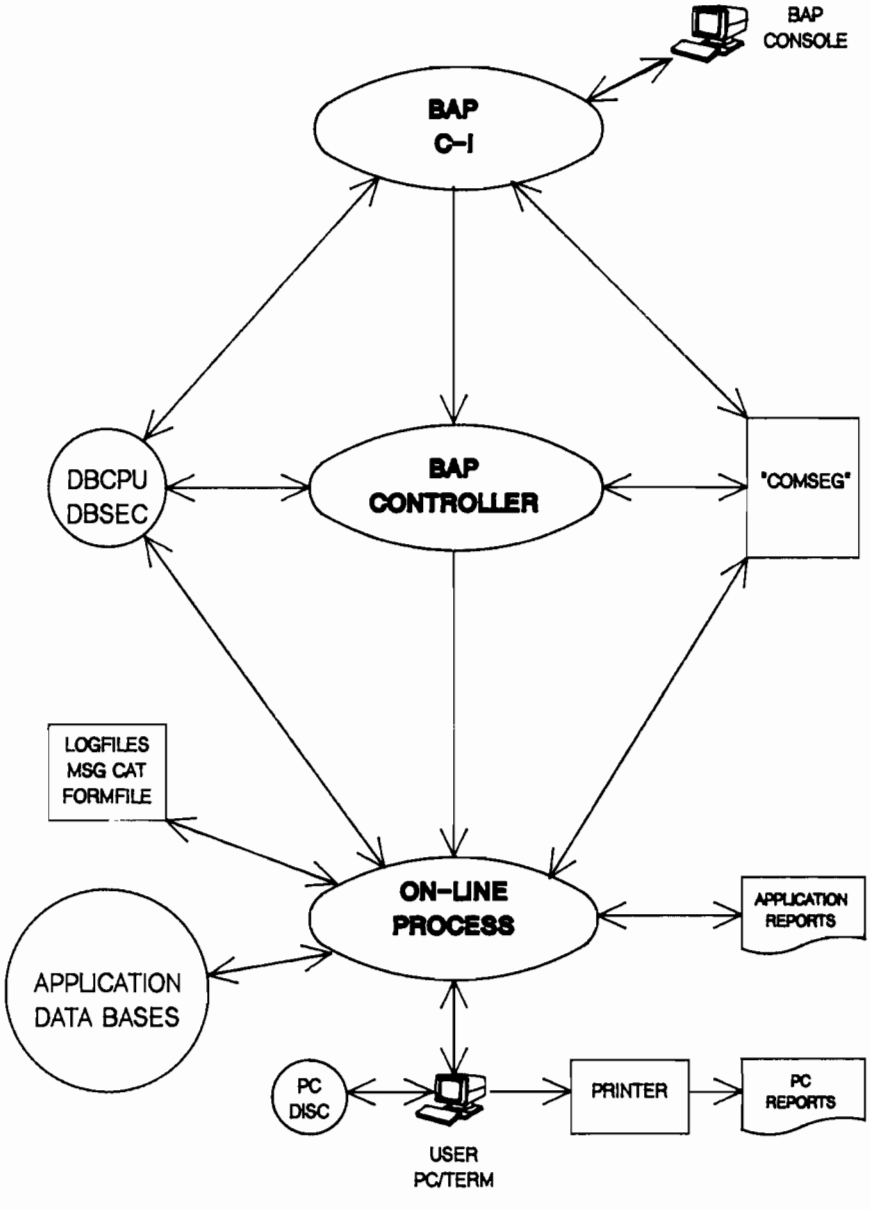


Figure 4 -- The Process Tree

IV. THE PROCESS TREE

Figure 4 shows the basic process tree designed to accomodate the aforementioned requirements, considerations and guidelines. For a more detailed description of the process tree and related performance issues, read "Using Process Handling to Optimize System Performance" in the Washington, D.C. INTEREX proceedings.

Following is an overview of each of the processes shown in Figure 4:

C.I. Process
The BAP Command Interpreter initializes logfiles, opens IPC files, get local RINS, creates the extra data segment "COMSEG", initializes the welcome message, and creates the Controller process. The process then does a timed read on the BAP console for requests for further actions such as: creating a user On-Line process, sending a message to a user, changing the welcome message, showing the status of all descendent processes, killing a process, telling a user process to terminate itself, pausing, and finally exiting the program.

Controller Process
The BAP Controller process gains access to "COMSEG", initializes variables and then suspends itself until it is awakened by the BAP C-I or a child process. When awakened, the BAP Controller scans "COMSEG" to determine what, if any, action it is to perform. The Controller may create a process, kill a process, send a message to the BAP C-I and will update "COMSEG" appropriately.

On-Line Process
The On-Line process gains access to "COMSEG", initializes data, opens files and data bases, opens a terminal for the user, and does a timed read on the terminal. If a command is entered, the appropriate application subprogram is called and the command and screen data passed to the subprogram. When the application subprogram is completed, control returns to the On-Line driver which checks "COMSEG" and takes any appropriate action. If the process is to continue, it loops back to a timed read on the user terminal. The On-Line process will be described in much greater detail in the remainder of this paper.

Functions Performed by Related Processes

As you saw in Section IV, several global functions were performed by processes other than the On-Line process. Among these were the following:

1. Initialization of "COMSEG", an inter-process extra data segment
2. Initialization of BAP logfiles.
3. Establishing communication links to remote CPU's.
4. Communication with the BAP console.
5. Initialization of local RIN's and IPC files.

If these functions were not external to the On-Line process, then they would have to be included in it. This would mean a much larger On-Line program in size with a correspondingly larger data stack. Thus the application subprograms would have to have a smaller limit on the size of its data stack. Remember that the maximum MAXDATA for a process is 31232. Thus, every word used by the driver is one word less that is available for the stack of the application subprogram. Figure 5 shows the impact of this.

Entity	BAP C-I	BAP Controller	BAP On-Line
Source Code Lines of Driver	2488	412	1677
Driver Initial Stack Size(w)	4366	174	2986

Figure 5 -- Driver Sizes

In summary, process handling was selected as a mechanism to control many terminals from a single process. This enabled us to satisfy Requirement 1 for no user MPE knowledge required. By transferring many of the initialization and control tasks to the C-I and Controller processes, the size and complexity of the On-Line process was reduced.

Functions Performed by the On-Line Process

The functions performed by the On-Line process are depicted in the following pseudo-code:

```
    <<***** Start of the On-Line Driver **********************>>

1.  Open entities shared by driver and application modules.
2.  Initialize data and buffers shared by the driver and the
    application subprograms.
3.  Sign on user verifying authorization to access data for
    the division specified. Retrieve a list of all commands
    for which the user is authorized to use in the division.

    <<***** Start of the Driver Read Terminal Loop ************>>

4.  Read the terminal.
5.  If a valid command was entered, then
6.     Log the transaction start.
7.     Call PCALnn passing shared data to the application module.

    <<***** Exit the Driver and enter PCALnn ******************>>

8.        PCALnn calls the appropriate application subprogram.

    <<***** Exit PCALnn and enter the application subprogram **>>

9.           Application subprogram executes its code.

    <<***** Return to PCALnn from application subprogram ******>>

10.       PCALnn returns to driver.

    <<***** Return to driver from PCALnn module ***************>>

11.     Log the transaction stop.
12. Check "COMSEG" to determine if driver should continue.
13. If not ok to continue go to step 15.
14. If user timeout, then go to step 3, else go to step 4.
15. Terminate process.
```

Figure 6 -- Functional Summary of the On-Line Process

VI. STRUCTURE OF THE ON-LINE APPLICATION PROGRAM

The following comments are relevant to the structure of the On-Line process.

1. The driver, OB', is in segment SEG' of the program.

2. Each command is assigned a command number. The driver computes what PCALnn routine to call from the command number. Each PCALnn routine is in a different segment.

3. The PCALnn subprogram computes what application subprogram to call in a CASE statement. Each of the 100 entries in the CASE statement has an application subprogram name. The PCALnn routine merely determines which one to execute.

   The reason that multiple PCALnn routines are required and that PCALnn routines are used at all is that a segment cannot have more that 256 externals. Since the number of application subprograms is several times greater than the aforementioned limit, the On-Line driver would not support direct calls to the application subprograms. Thus, several PCALnn subprograms were constructed, each which could call 100 application modules. This design would then support the ability to grow to many thousand subprograms; however, segment limits are sure to be met before that point. Finally, the structure sounds somewhat inefficient, but the one second response time indicates this is not significant.

4. The application subprograms are packed into segments in either the program itself or the group SL.

5. Low level subprograms (intrinsics and subprograms), which may be called by the application subprograms, are contained in the account SL. These routines are usually classed as intrinsics and created in conjunction with the BUILDINT program. Many of these low level routines are packed into each segment of the pub SL.

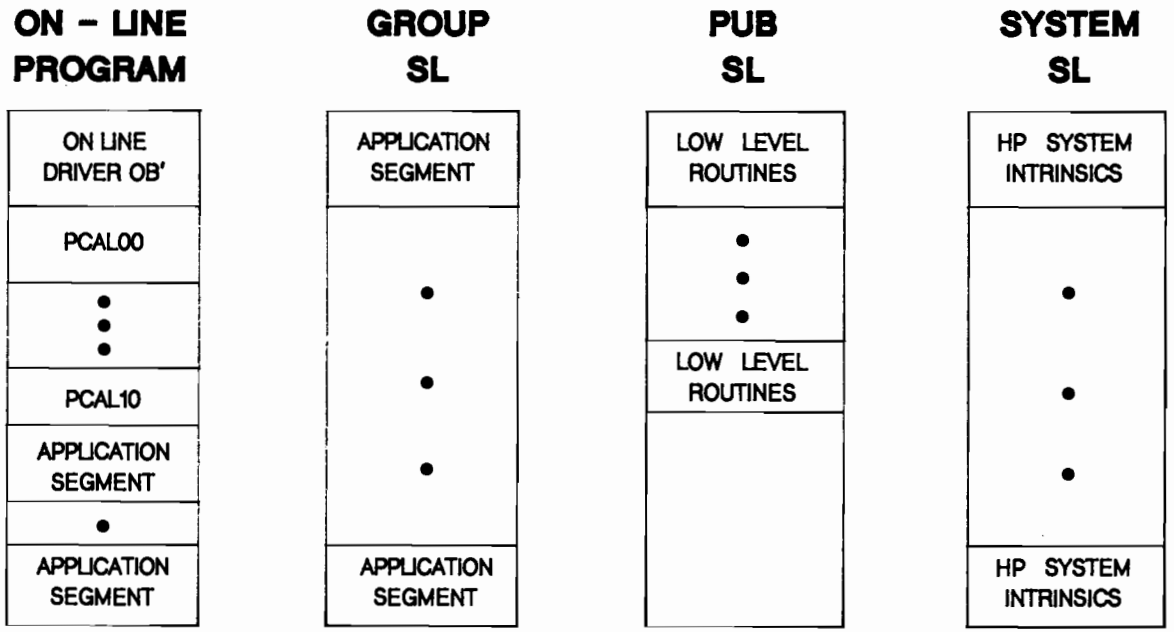


Figure 7 -- Segment Structure of the On - Line Process

VII. CHANGING / EXPANDING THE ON-LINE APPLICATION PROGRAM

As a standard operating procedure, the application program, the group SL and the pub SL are rebuilt each night from USL's. This is done by streaming three different jobs.

Building the Program
First, an empty program file and an empty USL file are built. The driver module, OB', and the PCALnn modules are copied from the existing driver USL into the new USL. The application subprograms are stored in two separate USL's since one USL won't hold them all (Thank's HP). The appropriate subprograms are copied into the new USL. When all the subprograms have be copied, they are then packed into segments using the NEWSEG command. The resultant new USL is then prepared into a program using the PREP command.

Building the Group SL
First, a new USL and a new SL are built. From each of the application code USL's, the appropriate subprograms are copied into the new USL. They are then repacked using the NEWSEG command. The packed segments are then added to the new SL using the ADDSL command.

Building the Pub SL
All low level procedures are in a single USL. Although change is infrequent, the pub SL is still rebuilt each night by first building a new SL and then simply adding the segments to the new SL using the ADDSL command.

Incorporating Application Code Changes
The revised application code is compiled into the appropriate application subprogram USL. The jobs to build the program file and group SL will install it correctly.

Installing New Application Commands and Subprograms
Installation of a new command is only slightly more difficult. The command is assigned a number and added to the security data base. The subprogram name is added to the PCALnn subprogram CASE statement related to the command number. The PCALnn source is then compiled into the driver USL. The application source is compiled into one of the two application USL's. The job stream for the program or group SL is then changed to copy and NEWSEG the subprogram. In the case of the group SL job, an ADDSL statement is also added. The corporate and divisional Controllers then authorize the appropriate users by adding the command to user's command list in the security data base.

Changing/Adding Low Level Routines
Changes or additions to the low level routines are simply compiled into the appropriate segment in the USL for the pub SL. The standard installation job to build a new pub SL will install it correctly.

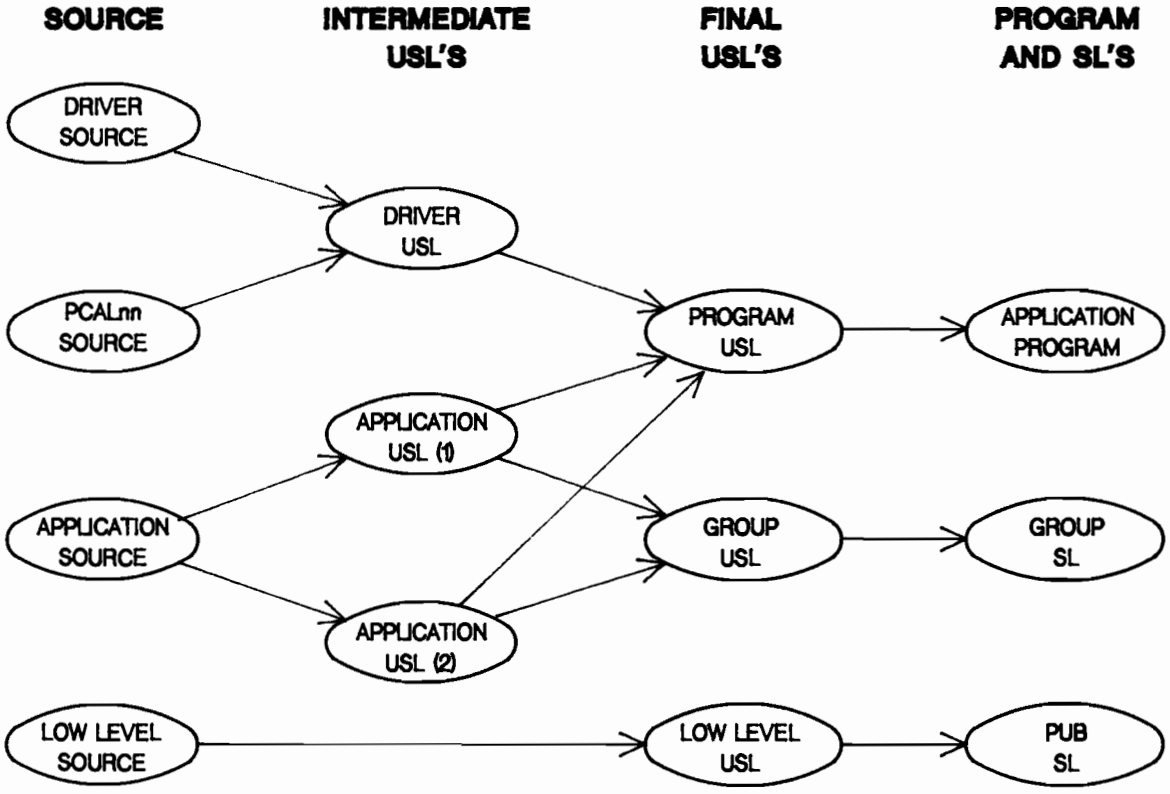


FIGURE 8 -- FLOW OF CODE DURING INSTALLATION

VIII. PERFORMANCE VS. DESIGN

If your still reading, now you get to the interesting part. The primary point of this paper is the using subprograms to improve performance. Obviously, if each subprogram was essentially identical to a stand alone program in code, then the only performance gain would be due to the lack of a process creation/termination each time the program was run. In our system, several additional things have been done to improve performance as follows:

1. An array of open data bases is saved in the driver and passed to the application subprogram for each command execution. When the application subprogram needs to access a data base, it checks to determine if it is in the array. If it is, then the data base can be used without opening it. Considering that the typical transaction may access five different data bases and the transaction count is 20,000 daily, this reduces dramatically the number of data base opens required.

   Not only are the data bases keep open and shared among the various application subprograms while a user is signed on to BAP, they are not closed when the user releases his terminal back to BAP. Since the SIGNON screen appears when the user signs off from BAP, the next user can access data bases which have been opened by a previous user. Remember that security is a function of the user/division/command, not the destination data base.

2. A similar gain is made with opening and initializing a user's terminal. Since several of our terminals are shared by several people, the ability to maintain a BAP process associated with the terminal, not a user means that neither a process has to be created nor a terminal open performed, both of which are costly operations in CPU and wall time.

3. All terminal reads are performed by the driver. The extensive error checking is thus common and independent from the application subprogram.

4. When a user enters a command, the command number has to be retrieved to determine if the user is authorized to use the command. The authorization for all commands executable by the user is maintained in the driver for that purpose. Thus, only one disc read per user signon to BAP is necessary to get the users security information. The command/command number cross reference is too large to keep in the driver without causing stack space problems for the subprogram; however, all is not lost.

A subset of the ten (10) most recently used commands and the associated information is maintained in the driver's stack. Since users typically perform a very small set of commands over and over, this technique eliminates approximately 90% of the I-O to the command set in the security data base.

5. Within the application subprogram, all data base access, V/3000 intrinsic calls and diagnostic error reporting calls are processed through a series of low level subroutines which have extensive error checking and error reporting provisions. This enables the tedious and lengthy code to be external to the application subprograms, which in turn usually provides for a smaller application subprogram. It also means that a global error analysis/reporting change can be implemented by changing only a single module in the pub SL, rather than by modifying hundreds of subprograms.

6. Measurement of transaction CPU and elapsed time is done by the driver. The transaction is logged by the driver. Thus all code associated with the transaction measurement and logging is separated from the application modules, again eliminating redundant code and facilitating ease of change.

The estimations show in Figure 9 are based on the following very conservative assumptions/calculations:

1. At least 1,000 process creations/terminations are prevented daily by having a single large program.

2. At least 10,000 data base opens are saved by sharing the data bases among users of the same process and among the various application subprograms which are executed within that process.

3. At least 1,000 V/3000 terminal opens/closes are saved.

4. Be having the employee's security information in a buffer in the driver and a list of most recently used commands, at least 120,000 I-O to the data base are saved.

CPU Saved	Function
50	1,000 Process Creations/Terminations
220	1,000 Terminal & Formfile Opens/Closes
1,000	10,000 Data Base Opens/Closes (UB delta 1)
240	120,000 Data Base Reads

Figure 9 -- Estimated Direct CPU Savings

IX. RESULTS: THE GOOD, THE BAD AND THE UGLY

The Good

The good news is that all of the original requirements have been met or exceeded. The most important is the one second response time which has been achieved.

The second bit of good news is the ease of installing global changes. Since all application modules return to the driver each transaction, it is very easy to install a new global housekeeping utility and have it effect all application code. This has turned out to be much more valuable than was anticipated, especially as the system has grown.

The Bad

There is no free ride. Training someone to understand how the pieces are put together is certainly more difficult with the program and SL structure. Thankfully, this need rarely arises.

One might assume that the configuration management associated with this software approach is more difficult than would be incurred with a more classical approach. I have come to doubt that assumption. Whenever a system is over a million lines of source and has a thousand application commands (tasks) which a user can execute, and has hundreds of users and terminals, the task of keeping track of all the pieces is going to be significant. The fact that all the above is accomplished with less than 500 work hours per year is a reasonable indication that the approach is amenable to control procedures.

The Ugly

Most of you will never see messages like "UNABLE TO OBTAIN SUFFICIENT DL STORAGE (LOAD ERR 71)" when you go to run a program and the LOADER balks. Or perhaps, such jewels as "ATTEMPT TO EXCEED MAXIMUM DIRECTORY SIZE" when you are adding a module to a USL. Since HP has been relatively effective in fixing bugs in the SEGMENTER, I'll be good and not make any other comments.

In essence, a number of brick walls have been crashed into by building a very large program, USL's and SL's. At this point, we have accepted the fact that this comes with the territory. Our software isn't bug free, so why should we expect HP's to be? We don't design software that can be expanded indefinitely, so why should we expect HP to?

In summary of "The Ugly", let me just say that if you are going to stretch the limits of MPE, then you should be prepared to occasionally find a limitation that isn't documented or isn't suppose to exist. The main thing is to not panic. After all, HP is sometimes very interested in your esoteric little problem. If not, then be persistent. If they still aren't interested (meaning they will do something other than take you name and phone number), then start looking for an alternate approach. After all, if you were dumb enough to get yourself into this fix, you will just have to be smart enough to get yourself out of it.

X. SUMMARY

Using a driver which calls many application code subprograms has provided a mechanism to provide excellent performance to a large and mature manufacturing/accounting/engineering system. In addition to performance gains, the approach provides an easy way to install global procedures which can affect all the application subprograms without changing the subprograms.

The approach requires a good understanding of the SEGMENTER and how program externals are satisfied. No special language is required.

By using the identical system for all applications, users throughout the corporation have become familiar with it. Training, therefore, is much easier during start-up of a new application area, because users are already familiar with the technique, on-line documentation and other facets of the system which are globally available to them.

Finally, by using the same approach in all application modules, a certain conformance to standards has been mandated to the software development team. Conversely, since the application software looks similar in structure regardless of the programmer, maintenance and support can more readily be supplied by any programmer.

Besides all that, it's been fun and exciting.

PROGRAMMATIC COMMUNICATION WITH A PC USING REFLECTION 3

George B. Scott
ELDEC CORPORATION
16700 13th Avenue West
Lynnwood, WA 98046-0100

I. INTRODUCTION

While I was preparing the abstract for this paper, I considered titles such as "There's No Fool Like an Old Fool", or "A Lesson in Humility". Both are apropos. If you are only interested in results, then go directly to Appendices A, B, and C, which contain the source for the low level routines used to transmit a file to and from a PC using the REFLECTION 3 software. If you are interested in why some of the source exists, or why it works at all, then you might enjoy reading the remainder of this paper which is outlined as follows:

II. THE CHALLENGE

One bright sunny day, a young man walked into my office and explained how one of our production facilities was considering installing carousel's for picking parts from stores for issue to production. For those of you unfamiliar with carousels, they are large units which have several tiers of shelves, each shelf with several bins, and each bin with several boxes, all of which rotate on a track. The location and quantity of the parts are maintained by a computer or PC. When requests for parts are entered, the carousel is rotated to bring the spot where the part exists to the person doing the picking.

In our environment, the carousel system being considered was going to be PC controlled. All that was needed was to transmit to the carousel a subset of the information that was normally printed on a pick list. The question was whether we, the Hewlett-Packard (HP) side of the house, could format and send a file to the PC and then pick up a file from the PC.

III. PLACING YOUR BET

Well, obviously we could do something that "simple". The fact that the user wanted to do this without exiting from our standard on-line application system certainly seemed reasonable. So after some discussion, the project was approved and a schedule was established.

To help you better understand the fun that follows, let me just say that things had been going quite good. A long string of successful development tasks had been completed and installed. Since I hadn't played too much with my IBM PC, I figured this was a good place to start. After all, I had used the HP3000 since 1974 and had figured out ways to transmit data to many different devices and systems. This shouldn't take more than a day or two. So the task of writing the application code to extract and format the data to be sent to the PC and the task of updating the HP from the data received back from the PC was assigned to one of our software development people.

I volunteered to write a generic low level routine that could be called by any process to transmit a file to or from a PC. After a little discussion, we decided that two routines would be better, one to transmit to the PC and one to receive from the PC. In any case, REFLECTION 3 software would be a part of this.

Now, I figured it would take the production people a few months to be ready to install their system, so we had plenty of time. Certainly, it would take several weeks to do the related applications programming. Being interested (meaning at long last, I had some new turf to play in), I decided to do my part now and have it ready long before it was needed. This was, perhaps, the only right decision I made for the rest of the journey, confirming the theory that ignorance is bliss.

IV. LEARNING TO READ

At this point, I had used the PC, which had been setting on my desk for several months, primarily as a terminal attached to our HP network. It was great. Thousands and thousands of lines of text and data could be displayed without wiping out any of it. At 5 PM, I could go back and see what I had done at 8 AM. This was great. When I had to use it as a PC, I just ask our resident PC expert what to do. Look at it this way, I didn't have anything to un-learn about the PC because I hadn't learned anything about it.

So I grabbed a copy of the REFLECTION software manual from an associate and started reading. I quickly discovered there was a file transfer function, so I decided to try that. I built a file on the HP, transferred it to the PC, went to the PC and reviewed it. Hey, this is really easy stuff. I transmitted the PC file back to the HP and checked it. No problem here, folks. At this rate, I figured I would be done by tomorrow.

Now, let's see how to do this programmaticly. After perusing the manual for awhile, it became apparent that there existed something called "REFLECTION Command Mode".

Note: Several years ago, I attended a course on user documentation. Dr. Ramey, who taught the class, explained how two different approaches were used by people reading documentation. The first type of person liked to start at the beginning and read to the end. This type also liked lots of narrative. The second type of person would sort of fan through a manual, look at pictures, diagrams and examples. Reading narrative was a last resort. I'm definitely a type 2.

Scanning the chapter, I quickly located the section on "Remote Use of Commands". Boy, I was really on a roll. It stated:

> "To complete the discussion of Reflection's command language, one more important use needs to be mentioned. The language can be used by a remote computer to control the operations of a PC. A remote user transmits instructions to a PC, which the PC, when programmed with Reflection, can understand and execute. The remote computer prefaces each command issued with the escape sequence ESC&oC, as follows:
>
> ESC&oC< Reflection command><carriage return>
>
> When reflection receives a command language command from the host computer, it transmits a response (technically a device completion code) when the command has been executed. The response is S for success or F for failure

and is transmitted as a Type 3 block transfer. (See page 349 for more information about block transfers.)"

I asked myself, "What could be simpler?" After reading the information related to block transfers on page 347, not page 349 as advertised, I concluded that I wasn't really interested at this point in more information about block transfers. Thus, the next thing to do was determine the sequence of commands necessary to send a file via command language to the PC.

Flipping through the manual, I quickly came to the section on "The File Transfer Subsystem". Aha! This is just what I wanted. I quickly discovered the existence of the "SEND" and the "RECEIVE" commands. Since the "RECEIVE" meant an HP to PC transmission, I decided I would henceforth have to consider the PC as "home", not my friendly HP3000. Certainly, that was a small sacrifice. At this point, I was sure I would be done by tomorrow.

The next step was obvious, simply execute the command and send a file to the PC. After executing the command, I transferred to the PC to look at the contents of my file. The only problem was that I couldn't find the file. Figuring that I must have deposited it in some other directory, I searched the PC. The file wasn't to be found. Oh well, I must have made an error somewhere. After repeating the process a couple of times, I decided to transmit a file from the PC to the HP. Still incurring the same results, I decided to talk a more learned user of REFLECTION.

The first comment from my associate was a question of why anyone would try to do this in a command mode rather than use the file transfer function and screens. So, I decided a demo was in order. I sat down at my associate's PC and proceeded to step through the various functions. I then was going to demonstrate that the file I had attempted to transfer was missing; however, it was there! So, I trotted over to our resident PC expert. He asks "What version of REFLECTION are you running?" By this time, everyone was heading home so we decided to research the problem tomorrow.

After discovering that the version of software installed on my PC didn't quite come close to matching the version of the documentation I had borrowed, we installed the latest version of REFLECTION. It worked perfect, just as advertised.

The point of this section is that it doesn't matter too much how you read. What matters is what you read. As I devoured my first piece of humble pie, I reflected on how many times I have mentioned to various people the importance of maintaining current software and documentation.

V. RULES OF THE GAME

After deciding there was little to be gained in berating myself further (No fun here, folks.), I decided to press onward.

Since the objective was to develop a generic procedure, I recalled that several variables were on the file transfer screen other than the source and destination files. In addition, the was a "config transfer" softkey which had a lot of options related to the file transfer. The various variables and options are summarized in Figure 1.

Range/Options	Default	Description
ASCII/BINARY	ASCII	Transfer Method
1 to 32766 bytes	512	Host Record Size
DELETE	-----	DELETE
APPEND	-----	APPEND
64-512	512	BLOCK-SIZE
YES/NO	YES	CR/LF=SEPARATOR
YES/NO	YES	CTRL-Z=EOF
YES/NO	YES	DELETE-TRAILING-SPACES
string	"RUN PCLINK.PUB.SYS"	HOST-STARTUP
1-9999	15	RECEIVE-TIMEOUT
1-9999	10	RETRY-LIMIT
YES/NO	YES	SEPARATOR=CR/LF
YES/NO	YES	SPACES-TO-TABS
YES/NO	YES	TABS-TO-SPACES
YES/NO	YES	WRITE-CTRL-Z

Figure 1 -- Typical Options for File Transfers

Note: ISO to Roman-8 is affected by both the PARITY and STRIP-TEXT parameters.

Obviously, a method to control these options was necessary. So, I returned to the manual. In the "Command Language Keywords" chapter, there was a description of the SET command, which seemed to cover all the necessary options. At this point,the only thing left was to write the program. Surely, this wouldn't take more than a few hours.

VI. LEARNING TO WRITE AND READ

Learning to Write

In addition to the information discussed in Section IV of this paper related to using commands to transfer files, the manual in the section on "Remote Use of Commands" further stated the following:

> "Note that while a PC programmed with Reflection executes instructions transmitted to it in this way, it is not itself operating in command mode; its own command line has not been accessed and no command has been entered at the keyboard."

The manual then proceeds to give an example of a TDP use file which transfers files.

So I decided to write a simple program to open the terminal as a file and write some commands to it as follows:

```
SET DISABLE-COMP-CODES YES
SET HOST-STARTUP "RUN PCLINK.PUB.SYS"
RECEIVE C:PCFILE.DAT FROM PCTOHP ASCII DELETE
```

Each of the above commands was preceded by the appropriate escape character sequence. The file wasn't transferred. I read the manual some more and tried a few variations on the above theme. No file was transferred to the PC. Now, you have to understand that I've known some of the folks at Walker, Richer & Quinn, Inc. ,hereafter referred to as WRQ, for several years. So to call and ask for help is definitely not good for the ego; but, being stuck was no fun either.

After a brief conversation with the folks at WRQ, it is obvious my problem is that I needed to create and activate the program PCLINK.PUB.SYS after sending the above commands. Well now, that piece of humble pie wasn't too big. So, I installed the create and activate, ran the program ran and the file was transferred. Eureka!

Learning to Read, the Second Time

Since the major objective (transferring a file programmatically) had been completed, the only thing left (or so I thought) was to make it pretty. Remembering the comments about completion codes, I decided it would be a good idea to test for success or failure on each command. So I changed the "SET DISABLE-COMPLETION-CODES" to "NO" and ran the program. The program brought up the REFLECTION transmission screen and proceeded to hang in a "Waiting for response from host" condition. I aborted the process and read some more in the manual. I reran the program and sure enough the PCLINK process had been created and activated before it became hung. Why would a process which was created and activated not run, but instead just hang, waiting for a response from the host?

The concept of calling WRQ twice in the same day for help was unbearable. I intently read the manual looking for a clue. I tried a couple of things. Nothing worked. At this point, I shut my door so I could have a frank discussion with the walls. Richard Nixon would have loved editing a tape of the conversation. I gave up and called WRQ again. Now it seems as though I was supposed to be reading the completion codes from a file called HPPCPORT. Furthermore, this file should be opened with the foptions %600 and the aoptions %404 with the device equal to the terminal.

Since I had turned on completion codes, the SET HOST-STARTUP command had sent one. The PCLINK process somehow detected that the HOST-STARTUP response hadn't been read and would not proceed.

Well, I admit I hadn't read each and every page of the manual, but I didn't recall seeing anything about "HPPCPORT" or any other information related to how these completion codes were to be intercepted. (Note: Since that time I have browsed through the entire manual. If anything about this is mentioned, it is certainly well disguised.) At this point, I opened and read the file HPPCPORT correctly, and again successfully transferred the file. Furthermore, the completion codes had been trapped and checked. Since I had incurred enough humbling in one day (asking for help twice is twice too much), I decided to continue on the following day, thus sacrificing my schedule of two days. Obviously, there couldn't be anything left that was tough.

The following day, I wrote a simple program to set and turn on all the options and test the resultant completion codes. I quickly learned that not every command returns a completion code. The easy way to circumvent hangs was to install a timed read. At least, I was then protected from waiting forever as I debugged my code.

At this point, the only thing I had left to do was write a version compatible with a process using V/3000.

VII. BLOCK MODE: THE CHICKEN AND THE EGG

Since the procedures I was developing were supposed to be low level, they were not to know what was happening in the calling procedure. Thus they would have to determine for themselves whether the terminal was in block mode.

I didn't remember a parameter in FGETINFO for this, but I checked it anyway. Unfortunately, I was right. Thinking perhaps FFILEINFO had a way to determine if the terminal was in block mode, I checked it; but, it wasn't there. As a last resort, I checked the FCONTROL options to see if it has some the ability to tell you that block mode was already turned on if you turned it on. No such luck was to prevail. Could I accept the cruelty of life and call WRQ again. After all, after being had twice, what's a third time?

I decided to call HP instead. It was explained to me that all I had to do was read the terminal status. How could I admit I didn't know what a terminal status was? After all, I was going to have to read the manual to decipher what values I had to analyze anyway.

So I grabbed a 2623 Terminal manual and proceeded. Sure enough, there was a section on terminal status. This would be child's play. Quickly ascertaining the escape sequence I needed to write, I coded the write and a timed read and proceeded. I decided to run the program in debug and check the write, then do the read and check it. The FWRITE succeeded. The FREAD hung. I broke and aborted. The next time I broke only after the FREAD. It was successful. Evidently, there is a rather sensitive timing situation between the FWRITE and the FREAD. Now all I had to do was to determine how to sense FORMAT MODE. I read the manual. I read chapters that listed all the escape sequences. No luck. I decided to cheat a little. Since the only application we had that used block mode were V/3000 applications, which always used FORMAT MODE, I just set the FORMAT MODE equal to the BLOCK MODE.

Finally, there was only PAGE MODE left. Since there was an appropriate bit in the terminal status data, I read and controlled PAGE MODE correctly. Now all I had to do was test it with a V/3000 application.

I decided to set the QUIET DISPLAY ON, QUIET STATUS ON and QUIET COMMAND ON and leave the DISABLE-COMP-CODE NO. By doing this, I would be able to detect the completion codes without having the anything written to a V/3000 screen if one was up. This didn't work. If you turn off the displays, etc, you can't get the completion codes. So I decided to add a requirement to the calling procedure. It would have to re-display the V/3000 screen and data if it had called the subroutine Note that with some further work, this could be done much more elegantly, and the user could be returned to the last line of his display prior to calling these procedures. Perhaps this will be completed by next year's conference.

So I turned off block, page and format modes; homed the cursor; and proceeded to execute the routine. Everything worked great. The REFLECTION screens which showed the progress of the file transfer were displayed, the commands and completion codes zipped across the bottom of the screen. The routine then reset appropriately the block, page and format modes, and returned to the calling program. Who could ask for more?

The calling program then homed the cursor, cleared the screen and put up the screen and data with VPUT, and set the cursor to the appropriate field. The only problem was the cursor wasn't in the correct spot and the data didn't all get to the screen. How long would this final inch take? After reflecting on the situation, I suspected that V/3000 had no knowledge that the terminal condition had changed since it had last been accessed and didn't check prior to doing the VPUT. I inserted a VINITFORM prior to the VPUT and everything worked.

Now, that wasn't so bad, was it?

X. SUMMARY

I hope that you will be considerate if you review the code in appendices A, B, and C. This is not meant to be exemplary examples of style, technique or anything other than how HP3000 programmatic communication with a PC can be done using REFLECTION 3.

Obviously, this little venture opens up many opportunities to do other things from within application programs running on the HP3000. We hope to continue our developments to provide a much more friendly integrated system for our users, so that they can accomplish their jobs whenever possible without having to transfer from one system to another.

I had a lot of fun writing this, certainly more fun than doing it. The lessons learned in technique were almost as valuable as the ones learned in humility. The fact that our shop delivered the software for the carousel project ahead of schedule makes it all seem worthwhile. After all that's the sort of thing that increases the size of the profit sharing check, which always makes me smile.

Finally, I hope you can use this rather crude start to expedite your journeys into integrating the HP3000 with PC's. Good luck on your journeys.

Appendix A -- Source for HP to PC File Transfer Subroutine

```
$TITLE" UT341S -- HPTOPC'XFER "
$CONTROL SUBPROGRAM,SEGMENT=SEG01,MAP
BEGIN
<<*************************************************************>>
<<NAME: UT341S            VERSION A00.00                 861022>>
<<                                                           >>
<<DESCRIPTION: SL ROUTINE FOR TRANSFERRING A FILE FROM HP TO PC>>
<<                                                           >>
<<AUTHOR: George B. Scott                                    >>
<<                                                           >>
<<REVISION      VERSION  COMMENTS                            >>
<<--------  ---  -------  --------                           >>
<<861022   GBS  A00.00   ORIGINAL.                           >>
<<*************************************************************>>
PROCEDURE HPTOPC'XFER(TERMFNUM,HPFNAME,PCFNAME,RETRYLIM,
                      RCVTIMEOUT,OPTS,STATUS);
   VALUE TERMFNUM,RETRYLIM,RCVTIMEOUT,OPTS;
   INTEGER TERMFNUM;
   ARRAY HPFNAME;
   ARRAY PCFNAME;
   INTEGER RETRYLIM;
   INTEGER RCVTIMEOUT;
   INTEGER OPTS;
   LOGICAL STATUS;
   OPTION CHECK 2;
    <<*************************************************************>>
    <<THIS ROUTINE SENDS A FILE FROM THE HP TO AN ATTACHED PC    >>
    <<INPUT:                                                     >>
    <<   TERMFNUM      - FNUM OF TERMINAL                       >>
    <<   HPFILE        - FILENAME OF FILE ON HP                 >>
    <<   PCFILE        - FILENAME OF FILE ON PC                 >>
    <<   RETRYLIM      - NUMBER OF RETRIES (LIMIT BY REFLECTIONS >>
    <<   RCVTIMEOUT    - TIMEOUT IN MILLISECONDS                >>
    <<   OPTS          - A BIT ARRAY AS FOLLOWS:                >>
    <<                   BIT 3     ISO-7 TO ROMAN-8            >>
    <<                                                         >>
    <<                   BIT 4     COMPLETION CODES ECHO OFF   >>
    <<                   BIT 5     SUPRESS CR/LF ON READS      >>
    <<                   BIT 6     DISABLE COMP CODES          >>
    <<                                                         >>
    <<                   BIT 7     SPACES TO TABS              >>
    <<                   BIT 8     DELETE TRAILING SPACES      >>
    <<                   BIT 9     CTRL-Z=EOF                  >>
    <<                                                         >>
    <<                   BIT 10    BINARY                      >>
    <<                   BIT 11    DELETE PREVIOUS FILE IF EXISTS>>
    <<                   BIT 12    SEPARATOR=CR/LF             >>
    <<                                                         >>
    <<                   BIT 13    QUIET STATUS                >>
    <<                   BIT 14    QUIET COMMAND               >>
    <<                   BIT 15    QUIET DISPLAY               >>
```

```
 <<OUTPUT:                                                          >>
 <<   STATUS  - 0 IF OK                                             >>
 <<*****************************************************************>>
BEGIN
BYTE ARRAY B'HPFNAME(*)=HPFNAME;
BYTE ARRAY B'PCFNAME(*)=PCFNAME;
LOGICAL
   BITON,
   BLOCKMODEOFF,
   BLOCKMODEON,
   COMPCODES,
   LDEV,
   MODE,
   NOARG,
   ORIG'BLKMODE,
   ORIG'ECHO,
   ORIG'FMTMODE,
   ORIG'PAGEMODE,
   READERR,
   SECS,
   SENDERR,
   WORD;
INTEGER
   BITNUM,
   DEVNUM,
   ERR,
   PARM,
   LEN,
   PORTFNUM,
   SONPIN,
   X,Y;
ARRAY PROGID(0:14);
   BYTE ARRAY B'PROGID(*)=PROGID;
ARRAY MSG(0:60);
   BYTE ARRAY B'MSG(*) = MSG;

INTRINSIC
   ACTIVATE,
   ASCII,
   COMMAND,
   CREATE,
   FCONTROL,
   FGETINFO,
   FOPEN,
   FREAD,
   FSETMODE,
   FWRITE,
   PAUSE,
   PRINTFILEINFO;
```

```
SUBROUTINE CLEARMSG;
BEGIN
   MOVE B'MSG:=" ";
   MOVE B'MSG(1):=B'MSG,(82);
END;

SUBROUTINE INIT'MSG;
BEGIN
   B'MSG:=%33;                          << ESC >>
   MOVE B'MSG(1):="&oC";
   MOVE B'MSG(4):=" ";
   MOVE B'MSG(5):=B'MSG(4),(73);
   B'MSG(78):=%15;                   <<CR>>
END;

SUBROUTINE READMSG;
BEGIN
   MODE:=4;
   SECS:=5;
   FCONTROL(PORTFNUM,MODE,SECS);
   LEN:=FREAD(PORTFNUM,MSG,-79);
   IF <> THEN
      BEGIN
      READERR:=TRUE;
      PRINTFILEINFO(PORTFNUM);
      END
   ELSE
      READERR:=FALSE;
END;

SUBROUTINE SENDMSG;
BEGIN
   FWRITE(TERMFNUM,MSG,-79,0);
   IF <> THEN
      BEGIN
      SENDERR:=TRUE;
      PRINTFILEINFO(TERMFNUM);
      END
   ELSE
      SENDERR:=FALSE;
END;

SUBROUTINE TESTBIT(WORD,BITNUM,BITON);
   LOGICAL WORD;
   INTEGER BITNUM;
   LOGICAL BITON;
BEGIN
   X:=0;
   X:=WORD & LSL(BITNUM);
   IF X<0 THEN
      BITON:=TRUE
```

```
   ELSE
      BITON:=FALSE;
END;

<<INITIALIZATION -- OPEN FILES ETC                                    >>
<<FIRST ISSUE FEQ FOR HPPCPORT TO APPROPRIATE TERM DEV                >>

STATUS:=0;
MOVE B'MSG:="FILE HPPCPORT;DEV=000   ";
FGETINFO(TERMFNUM,,,,,,LDEV);
DEVNUM:=INTEGER(LDEV);
LEN:=ASCII(DEVNUM,-10,B'MSG(20));
B'MSG(21):=%15;  << CR >>
COMMAND(B'MSG,ERR,PARM);
IF <> THEN STATUS := STATUS LOR %1;

<< OPEN THE FILE HPPCPORT TO READ MSGS FROM REFLECTIONS              >>
MOVE B'MSG:="HPPCPORT;";
PORTFNUM:=FOPEN(B'MSG,%600,%404,,B'MSG(18));
IF <> THEN STATUS := STATUS LOR %2;

<<******* TURN OFF ECHO IF NECESSARY & SUPRESS CR/LF                 >>
BITNUM:=4;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   FCONTROL(TERMFNUM,13,ORIG'ECHO); << TURN ECHO OFF,SAVE ORIG>>
BITNUM:=5;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   FSETMODE(PORTFNUM,%4);  <<SUPPRESS CR,LF ON READS>>

<<******* SAVE CURRENT STATUS OF BLK, FMT AND PAGE MODES             >>
CLEARMSG;
MSG    := %15536; <<ESC caret >>
LEN:=-2;
FWRITE(TERMFNUM,MSG,LEN,%320);
IF <> THEN STATUS:=STATUS LOR %2;
CLEARMSG;
LEN:=FREAD(TERMFNUM,MSG,-10);
IF MSG = %15534 THEN          << ESC \  >>
   BEGIN
   ORIG'FMTMODE:=0;
   ORIG'BLKMODE:=0;
   ORIG'PAGEMODE:=0;
   ORIG'PAGEMODE:=MSG(1).(12:1);
   ORIG'BLKMODE :=MSG(2).(14:1);
   ORIG'FMTMODE :=ORIG'BLKMODE;  <<HELL OF AN ASSUMPTION >>
   END
ELSE
   GOTO BAILOUT;
```

```
<<NOW THAT ORIG BLK,FMT AND PAGE MODES ARE SAVED, TURN THEM OFF>>
<< TURN OFF BLK, FMT AND PAGE MODES                              >>
CLEARMSG;
MOVE B'MSG:=" X &s0D &k0B";
B'MSG:=%33; <<ESC>>
B'MSG(2):=%33;
B'MSG(7):=%33;
SENDMSG;
IF SENDERR THEN STATUS:=STATUS LOR %4;

MOVE B'PROGID:="PCLINK.PUB.LIB  ";
<<********** END OF INITIALIZATION STAGE ********************>>

INIT'MSG;
BITNUM:=6;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   BEGIN
   COMPCODES:=FALSE;
   MOVE B'MSG(4):="SET DISABLE-COMP-CODES YES";
   END
ELSE
   BEGIN
   COMPCODES:=TRUE;
   MOVE B'MSG(4):="SET DISABLE-COMP-CODES NO";
   END;
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %10;
INIT'MSG;
BITNUM:=14;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="QUIET COMMAND ON"
ELSE
   MOVE B'MSG(4):="QUIET COMMAND OFF";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %20;
INIT'MSG;
BITNUM:=13;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="QUIET STATUS ON"
ELSE
   MOVE B'MSG(4):="QUIET STATUS OFF";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
```

```
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %40;
INIT'MSG;
BITNUM:=15;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="QUIET DISPLAY ON"
ELSE
   MOVE B'MSG(4):="QUIET DISPLAY OFF";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %100;
INIT'MSG;
MOVE B'MSG(4):="SET HOST-STARTUP *RUN PCLINK.PUB.LIB*";
X:=4;
WHILE X<84 DO
   BEGIN
   IF B'MSG(X)="*" THEN B'MSG(X):=%42;
   X:=X+1;
   END;
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %200;
INIT'MSG;
MOVE B'MSG(4):="SET RECEIVE-TIMEOUT ";
IF RCVTIMEOUT < 0 OR RCVTIMEOUT > 9999 THEN
   RCVTIMEOUT := 10;
LEN:=ASCII(RCVTIMEOUT,10,B'MSG(24));
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
INIT'MSG;
MOVE B'MSG(4):="SET RETRY-LIMIT ";
IF RETRYLIM < 0 OR RETRYLIM > 9999 THEN
   RETRYLIM := 10;
LEN:=ASCII(RETRYLIM,10,B'MSG(20));
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %400;
INIT'MSG;
BITNUM:=12;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="SET SEPARATOR=CR/LF YES"
ELSE
   MOVE B'MSG(4):="SET SEPARATOR=CR/LF NO";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
```

```
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %1000;
INIT'MSG;
BITNUM:=7;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="SET SPACES-TO-TABS YES"
ELSE
   MOVE B'MSG(4):="SET SPACES-TO-TABS NO";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %2000;
INIT'MSG;
BITNUM:=9;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="SET CTRL-Z=EOF YES"
ELSE
   MOVE B'MSG(4):="SET CTRL-Z=EOF NO";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %4000;
INIT'MSG;
BITNUM:=8;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="SET DELETE-TRAILING-SPACES YES"
ELSE
   MOVE B'MSG(4):="SET DELETE-TRAILING-SPACES NO";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %10000;
INIT'MSG;
MOVE B'MSG(4):="RECEIVE ";
X:=0;
WHILE B'PCFNAME(X)<>" " DO
   BEGIN
   MOVE B'MSG(12+X):=B'PCFNAME(X),(1);
   X:=X+1;
   END;
MOVE B'MSG(12+X):=" FROM ";
Y:=X+18;
X:=0;
WHILE B'HPFNAME(X)<>" "  AND B'HPFNAME(X)<>";" DO
   BEGIN
   MOVE B'MSG(Y+X):=B'HPFNAME(X),(1);
   X:=X+1;
   END;
Y:=Y+X+1;
```

```
   BITNUM:=10;
   TESTBIT(OPTS,BITNUM,BITON);
   IF BITON = TRUE THEN
      MOVE B'MSG(Y):="BINARY "
   ELSE
      MOVE B'MSG(Y):="ASCII  ";
   IF BITON THEN
      Y:=Y+7
   ELSE
      Y:=Y+6;
   BITNUM:=11;
   TESTBIT(OPTS,BITNUM,BITON);
   IF BITON = TRUE THEN
      MOVE B'MSG(Y):="DELETE ";
   SENDMSG;
   CLEARMSG;
   IF COMPCODES THEN READMSG;
   IF COMPCODES AND B'MSG<>"R" THEN STATUS:=STATUS LOR %20000;
   << ********** CREATE PROGRAM PCLINK.PUB.LIB TO TRANSMIT DATA >>
   CREATE(B'PROGID,,SONPIN,,%1);  <<SUSPEND AFTER CREATION>>
   IF <> THEN STATUS:=STATUS LOR %10;
   ACTIVATE(SONPIN,2);  << AWAKENED BY SON>>
   IF <> THEN STATUS:=STATUS LOR %100000;
   IF COMPCODES THEN READMSG;
   IF COMPCODES AND B'MSG<>"S" THEN
      BEGIN
      STATUS:=STATUS LOR %40000;
      <<RESET HANDSHAKE>>
      CLEARMSG;
      MSG:=%010415;  <<DC1, CR --- RESET HANDSHAKE >>
      SENDMSG;
      END;
   <<******** RESET BLK, FMT AND PAGE MODES TO ORIGINAL *********>>
   CLEARMSG;
   MOVE B'MSG:=" X &s0D &k0B";
   B'MSG:=%33; <<ESC>>
   B'MSG(2):=%33;
   B'MSG(7):=%33;
   IF ORIG'FMTMODE = 1 THEN MOVE B'MSG(1):="W";
   IF ORIG'PAGEMODE= 1 THEN MOVE B'MSG(5):="1";
   IF ORIG'BLKMODE = 1 THEN MOVE B'MSG(10):="1";
   SENDMSG;

BAILOUT:

   <<****** TURN ON ECHO TO OBSERVE REFLECTIONS SCREEN          >>
   IF ORIG'ECHO = 0 THEN
      FCONTROL(TERMFNUM,12,ORIG'ECHO); << TURN ECHO ON>>
   BITNUM:=5;
   TESTBIT(OPTS,BITNUM,BITON);
```

```
   IF BITON THEN
      FSETMODE(PORTFNUM,%0);  <<RE-ENABLE CR,LF ON READS>>


END;
END.  <<END OF HPTOPC'XFER>>
```

Appendix B -- Source for PC to HP File Transfer Subroutine

```
$TITLE" UT342S -- PCTOHP'XFER "
$CONTROL SUBPROGRAM,SEGMENT=SEG01,MAP
BEGIN
<<*****************************************************************>>
<<NAME: UT342S            VERSION A00.00                  861031>>
<<                                                                >>
<<DESCRIPTION: SL ROUTINE FOR TRANSFERRING A FILE FROM PC TO HP>>
<<                                                                >>
<<AUTHOR: George B. Scott                                         >>
<<                                                                >>
<<REVISION      VERSION  COMMENTS                                 >>
<<--------- ---  -------  --------                                 >>
<<861031   GBS  A00.00   ORIGINAL.                                >>
<<*****************************************************************>>
PROCEDURE PCTOHP'XFER(TERMFNUM,HPFNAME,PCFNAME,RETRYLIM,
                      RCVTIMEOUT,RECSIZE,OPTS,STATUS);
   VALUE TERMFNUM,RETRYLIM,RCVTIMEOUT,RECSIZE,OPTS;
   INTEGER TERMFNUM;
   ARRAY HPFNAME;
   ARRAY PCFNAME;
   INTEGER RETRYLIM;
   INTEGER RCVTIMEOUT;
   INTEGER RECSIZE;
   INTEGER OPTS;
   LOGICAL STATUS;
   OPTION CHECK 2;
    <<*****************************************************************>>
    <<THIS ROUTINE SENDS A FILE FROM THE HP TO AN ATTACHED PC     >>
    <<INPUT:                                                      >>
    <<   TERMFNUM      - FNUM OF TERMINAL                        >>
    <<   HPFILE        - FILENAME OF FILE ON HP                  >>
    <<   PCFILE        - FILENAME OF FILE ON PC                  >>
    <<   RETRYLIM      - NUMBER OF RETRIES (LIMIT BY REFLECTIONS >>
    <<   RCVTIMEOUT    - TIMEOUT IN MILLISECONDS                 >>
    <<   RECSIZE       - SIZE IN BYTES (POSITIVE)                >>
    <<   OPTS          - A BIT ARRAY AS FOLLOWS:                 >>
    <<                    BIT 3     ISO-7 TO ROMAN-8            >>
    <<                                                          >>
    <<                    BIT 4     COMPLETION CODES ECHO OFF   >>
    <<                    BIT 5     SUPRESS CR/LF ON READS      >>
    <<                    BIT 6     DISABLE COMP CODES          >>
    <<                                                          >>
    <<                    BIT 7     SPACES TO TABS              >>
    <<                    BIT 8     undefined                   >>
    <<                    BIT 9     CTRL-Z=EOF                  >>
    <<                                                          >>
    <<                    BIT 10    BINARY                      >>
    <<                    BIT 11    DELETE PREVIOUS FILE IF EXISTS>>
    <<                    BIT 12    SEPARATOR=CR/LF             >>
    <<                                                          >>
    <<                    BIT 13    QUIET STATUS                >>
```

```
 <<                    BIT 14    QUIET COMMAND                 >>
 <<                    BIT 15    QUIET DISPLAY                 >>
 <<OUTPUT:                                                      >>
 <<   STATUS  - 0 IF OK                                         >>
 <<*************************************************************>>
BEGIN
BYTE ARRAY B'HPFNAME(*)=HPFNAME;
BYTE ARRAY B'PCFNAME(*)=PCFNAME;
LOGICAL
   BITON,
   BLOCKMODEOFF,
   BLOCKMODEON,
   COMPCODES,
   LDEV,
   MODE,
   NOARG,
   ORIG'BLKMODE,
   ORIG'ECHO,
   ORIG'FMTMODE,
   ORIG'PAGEMODE,
   READERR,
   SECS,
   SENDERR,
   WORD;
INTEGER
   BITNUM,
   DEVNUM,
   ERR,
   PARM,
   LEN,
   PORTFNUM,
   SONPIN,
   X,Y;
ARRAY PROGID(0:14);
   BYTE ARRAY B'PROGID(*)=PROGID;
ARRAY MSG(0:60);
   BYTE ARRAY B'MSG(*) = MSG;

INTRINSIC
   ACTIVATE,
   ASCII,
   COMMAND,
   CREATE,
   FCONTROL,
   FGETINFO,
   FOPEN,
   FREAD,
   FSETMODE,
   FWRITE,
   PAUSE,
   PRINTFILEINFO;
```

```
SUBROUTINE CLEARMSG;
BEGIN
   MOVE B'MSG:=" ";
   MOVE B'MSG(1):=B'MSG,(82);
END;

SUBROUTINE INIT'MSG;
BEGIN
   B'MSG:=%33;                          << ESC >>
   MOVE B'MSG(1):="&oC";
   MOVE B'MSG(4):=" ";
   MOVE B'MSG(5):=B'MSG(4),(73);
   B'MSG(78):=%15;                    <<CR>>
END;

SUBROUTINE READMSG;
BEGIN
   MODE:=4;
   SECS:=5;
   FCONTROL(PORTFNUM,MODE,SECS);
   LEN:=FREAD(PORTFNUM,MSG,-79);
   IF <> THEN
      BEGIN
      READERR:=TRUE;
      PRINTFILEINFO(PORTFNUM);
      END
   ELSE
      READERR:=FALSE;
END;

SUBROUTINE SENDMSG;
BEGIN
   FWRITE(TERMFNUM,MSG,-79,0);
   IF <> THEN
      BEGIN
      SENDERR:=TRUE;
      PRINTFILEINFO(TERMFNUM);
      END
   ELSE
      SENDERR:=FALSE;
END;

SUBROUTINE TESTBIT(WORD,BITNUM,BITON);
   LOGICAL WORD;
   INTEGER BITNUM;
   LOGICAL BITON;
BEGIN
   X:=0;
   X:=WORD & LSL(BITNUM);
   IF X<0 THEN
      BITON:=TRUE
```

```
   ELSE
      BITON:=FALSE;
END;

<<INITIALIZATION -- OPEN FILES ETC                                    >>
<<FIRST ISSUE FEQ FOR HPPCPORT TO APPROPRIATE TERM DEV                >>

STATUS:=0;
MOVE B'MSG:="FILE HPPCPORT;DEV=000   ";
FGETINFO(TERMFNUM,,,,,,LDEV);
DEVNUM:=INTEGER(LDEV);
LEN:=ASCII(DEVNUM,-10,B'MSG(20));
B'MSG(21):=%15;  << CR >>
COMMAND(B'MSG,ERR,PARM);
IF <> THEN STATUS := STATUS LOR %1;

<< OPEN THE FILE HPPCPORT TO READ MSGS FROM REFLECTIONS              >>
MOVE B'MSG:="HPPCPORT;";
PORTFNUM:=FOPEN(B'MSG,%600,%404,,B'MSG(18));
IF <> THEN STATUS := STATUS LOR %2;

<<******* TURN OFF ECHO IF NECESSARY & SUPRESS CR/LF                 >>
BITNUM:=4;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   FCONTROL(TERMFNUM,13,ORIG'ECHO); << TURN ECHO OFF,SAVE ORIG>>
BITNUM:=5;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   FSETMODE(PORTFNUM,%4);  <<SUPPRESS CR,LF ON READS>>

<<******* SAVE CURRENT STATUS OF BLK, FMT AND PAGE MODES             >>
CLEARMSG;
MSG    := %15536; <<ESC caret >>
LEN:=-2;
FWRITE(TERMFNUM,MSG,LEN,%320);
IF <> THEN STATUS:=STATUS LOR %2;
CLEARMSG;
LEN:=FREAD(TERMFNUM,MSG,-10);
IF MSG = %15534 THEN          << ESC \  >>
   BEGIN
   ORIG'FMTMODE:=0;
   ORIG'BLKMODE:=0;
   ORIG'PAGEMODE:=0;
   ORIG'PAGEMODE:=MSG(1).(12:1);
   ORIG'BLKMODE :=MSG(2).(14:1);
   ORIG'FMTMODE :=ORIG'BLKMODE;  <<HELL OF AN ASSUMPTION >>
   END
ELSE
   GOTO BAILOUT;
```

```
<<NOW THAT ORIG BLK,FMT AND PAGE MODES ARE SAVED, TURN THEM OFF>>
<< TURN OFF BLK, FMT AND PAGE MODES                              >>
CLEARMSG;
MOVE B'MSG:=" X &s0D &k0B";
B'MSG:=%33; <<ESC>>
B'MSG(2):=%33;
B'MSG(7):=%33;
SENDMSG;
IF SENDERR THEN STATUS:=STATUS LOR %4;

MOVE B'PROGID:="PCLINK.PUB.LIB  ";
<<********** END OF INITIALIZATION STAGE ********************>>

INIT'MSG;
BITNUM:=6;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   BEGIN
   COMPCODES:=FALSE;
   MOVE B'MSG(4):="SET DISABLE-COMP-CODES YES";
   END
ELSE
   BEGIN
   COMPCODES:=TRUE;
   MOVE B'MSG(4):="SET DISABLE-COMP-CODES NO";
   END;
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %10;
INIT'MSG;
BITNUM:=14;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="QUIET COMMAND ON"
ELSE
   MOVE B'MSG(4):="QUIET COMMAND OFF";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %20;
INIT'MSG;
BITNUM:=13;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="QUIET STATUS ON"
ELSE
   MOVE B'MSG(4):="QUIET STATUS OFF";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
```

```
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %40;
INIT'MSG;
BITNUM:=15;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="QUIET DISPLAY ON"
ELSE
   MOVE B'MSG(4):="QUIET DISPLAY OFF";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %100;
INIT'MSG;
MOVE B'MSG(4):="SET HOST-STARTUP *RUN PCLINK.PUB.LIB*";
X:=4;
WHILE X<84 DO
   BEGIN
   IF B'MSG(X)="*" THEN B'MSG(X):=%42;
   X:=X+1;
   END;
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %200;
INIT'MSG;
MOVE B'MSG(4):="SET RECEIVE-TIMEOUT ";
IF RCVTIMEOUT < 0 OR RCVTIMEOUT > 9999 THEN
   RCVTIMEOUT := 10;
LEN:=ASCII(RCVTIMEOUT,10,B'MSG(24));
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
INIT'MSG;
MOVE B'MSG(4):="SET RETRY-LIMIT ";
IF RETRYLIM < 0 OR RETRYLIM > 9999 THEN
   RETRYLIM := 10;
LEN:=ASCII(RETRYLIM,10,B'MSG(20));
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %400;
INIT'MSG;
BITNUM:=12;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="SET SEPARATOR=CR/LF YES"
ELSE
   MOVE B'MSG(4):="SET SEPARATOR=CR/LF NO";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
```

```
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %1000;
INIT'MSG;
BITNUM:=7;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="SET SPACES-TO-TABS YES"
ELSE
   MOVE B'MSG(4):="SET SPACES-TO-TABS NO";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %2000;
INIT'MSG;
BITNUM:=9;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON THEN
   MOVE B'MSG(4):="SET CTRL-Z=EOF YES"
ELSE
   MOVE B'MSG(4):="SET CTRL-Z=EOF NO";
SENDMSG;
CLEARMSG;
IF COMPCODES THEN READMSG;
IF COMPCODES AND B'MSG<>"S" THEN STATUS:=STATUS LOR %4000;
INIT'MSG;
MOVE B'MSG(4):="CONTINUE ON";
SENDMSG;
IF COMPCODES THEN READMSG;
INIT'MSG;
MOVE B'MSG(4):="SEND ";
X:=0;
WHILE B'PCFNAME(X)<>" " DO
   BEGIN
   MOVE B'MSG(9+X):=B'PCFNAME(X),(1);
   X:=X+1;
   END;
MOVE B'MSG(9+X):=" TO ";
Y:=X+13;
X:=0;
WHILE B'HPFNAME(X)<>" "  AND B'HPFNAME(X)<>";" DO
   BEGIN
   MOVE B'MSG(Y+X):=B'HPFNAME(X),(1);
   X:=X+1;
   END;
Y:=Y+X+1;
BITNUM:=10;
TESTBIT(OPTS,BITNUM,BITON);
IF BITON = TRUE THEN
   MOVE B'MSG(Y):="BINARY "
ELSE
   MOVE B'MSG(Y):="ASCII  ";
IF BITON THEN
```

```
      Y:=Y+7
   ELSE
      Y:=Y+6;
   BITNUM:=11;
   TESTBIT(OPTS,BITNUM,BITON);
   IF BITON = TRUE THEN
      BEGIN
      MOVE B'MSG(Y):="DELETE ";
      Y:=Y+7;
      END;
   IF RECSIZE <> 0 THEN
      BEGIN
      MOVE B'MSG(Y):="REC = ";
      Y:=Y+6;
      LEN:=ASCII(RECSIZE,10,B'MSG(Y));
      END;
   SENDMSG;
   CLEARMSG;
   IF COMPCODES THEN READMSG;
   IF COMPCODES AND B'MSG<>"R" THEN STATUS:=STATUS LOR %20000;
   IF B'MSG = "R" THEN
      BEGIN
      << **** CREATE PROGRAM PCLINK.PUB.LIB TO TRANSMIT DATA >>
      CREATE(B'PROGID,,SONPIN,,%1);  <<SUSPEND AFTER CREATION>>
      IF <> THEN STATUS:=STATUS LOR %10;
      ACTIVATE(SONPIN,2);  << AWAKENED BY SON>>
      IF <> THEN STATUS:=STATUS LOR %100000;
      IF COMPCODES THEN READMSG;
      END;
   IF COMPCODES AND B'MSG<>"S" THEN
      BEGIN
      STATUS:=STATUS LOR %40000;
      <<RESET HANDSHAKE>>
      CLEARMSG;
      MSG:=%010415;  <<DC1, CR --- RESET HANDSHAKE >>
      SENDMSG;
      END;
   <<******* RESET BLK, FMT AND PAGE MODES TO ORIGINAL *********>>
   CLEARMSG;
   MOVE B'MSG:=" X &s0D &k0B";
   B'MSG:=%33; <<ESC>>
   B'MSG(2):=%33;
   B'MSG(7):=%33;
   IF ORIG'FMTMODE = 1 THEN MOVE B'MSG(1):="W";
   IF ORIG'PAGEMODE= 1 THEN MOVE B'MSG(5):="1";
   IF ORIG'BLKMODE = 1 THEN MOVE B'MSG(10):="1";
   SENDMSG;

BAILOUT:
```

```
   <<****** TURN ON ECHO TO OBSERVE REFLECTIONS SCREEN          >>
   IF ORIG'ECHO = 0 THEN
      FCONTROL(TERMFNUM,12,ORIG'ECHO); << TURN ECHO ON>>
   BITNUM:=5;
   TESTBIT(OPTS,BITNUM,BITON);
   IF BITON THEN
      FSETMODE(PORTFNUM,%0);  <<RE-ENABLE CR,LF ON READS>>


END;
END.  <<END OF PCTOHP'XFER>>
```

APPENDIX C -- Source for program which calls HPTOPC'XFER

```
$TITLE"MLTESTS"
$CONTROL SOURCE,MAP
BEGIN
<<*****************************************************************>>
<<NAME: MLTESTS          VERSION A00.00                      870507>>
<<                                                                 >>
<<DESCRIPTION: SAMPLE PROGRAM CALLING HPTOPC'XFER                   >>
<<             Note: PREP with IA,BA,PH capabilities               >>
<<AUTHOR: George B. Scott                                          >>
<<                                                                 >>
<<REVISION      VERSION  COMMENTS                                  >>
<<--------- ---  -------  --------                                  >>
<<870507   GBS  A00.00   ORIGINAL.                                 >>
<<*****************************************************************>>

ARRAY BUF(0:10);
   BYTE ARRAY B'BUF(*)=BUF;
ARRAY FNAME(0:20);
   BYTE ARRAY B'FNAME(*)=FNAME;
ARRAY PCNAME(0:20);
ARRAY HPNAME(0:20);
INTEGER FNUM,RTRY,RTIMOUT,ROPTS,LEN;
LOGICAL STATUS;
INTRINSIC FOPEN,QUIT,PRINT,ASCII;

PROCEDURE HPTOPC'XFER(TERMFNUM,HPB'FNAME,PCB'FNAME,RETRYLIM,
                      RCVTIMEOUT,OPTS,STATUS);
   VALUE TERMFNUM,RETRYLIM,RCVTIMEOUT,OPTS;
   INTEGER TERMFNUM;
   ARRAY HPB'FNAME;
   ARRAY PCB'FNAME;
   INTEGER RETRYLIM;
   INTEGER RCVTIMEOUT;
   INTEGER OPTS;
   LOGICAL STATUS;
   OPTION EXTERNAL,CHECK 2;

FNUM:=FOPEN(,%614,%404);
IF <> THEN QUIT(1);
MOVE PCNAME:="C:TOPC01.DAT ";
MOVE HPNAME:="TOPC01.LASVEGAS "; <<Needs to exist on HP>>
RTRY:=10;
RTIMOUT:=15;
ROPTS:=%000120;
STATUS:=0;
HPTOPC'XFER(FNUM,HPNAME,PCNAME,RTRY,RTIMOUT,ROPTS,STATUS);
MOVE BUF:="                              ";
LEN:=ASCII(STATUS,8,B'BUF(10));
PRINT(BUF,-20,0);

END.  <<END OF MLTESTS>>
```

Structured Program To Program Communication

One Way To Do It !

by

Malcolm Sharratt

Hewlett-Packard Company
Office Productivity Division
Nine Mile Ride
Wokingham
Berkshire
Pinewood
England

SUMMARY

Communication between programs is a necessary part of many software products. It is surprising then that few, if any, people have attempted to give a structure to the way in which this frequently occurring task is performed.

A requirement for a large number of programs to communicate freely with each other was identified as part of the recently completed enhancements to HPDeskManager. Rather than let each program communicate in its individual way, a set of rules was developed to guide the author of each subsystem program towards achieving a standard method of inter process communication.

Although this method was developed specifically to facilitate the integration of subsystems into the HPDeskManager product, the design is flexible enough to be used in any project requiring programs to communicate data. If the requirement is to frequently produce programs which communicate, use of this method can save time by allowing the inclusion of standard procedures into each program.

This paper outlines the requirement identified for the HPDeskManager product. It then details the way in which the problem was solved by the development of a protocol to govern the program to program communication. Each of the elements of the protocol is fully explained in order to give the reader a starting point should he wish to implement a similar standard.

INTRODUCTION

As part of the recent major HPDeskManager enhancements it was seen as necessary to link all of the components of the Personal Productivity Center (PPC) closer together. This required HPDeskManager, the hub of the PPC, to be able to read, edit, print and convert files of many varying formats. The integration of specialized converters as well as other PPC components (such as HPListKeeper) was needed and they all required information from HPDeskManager at execution time. To simplify this process a set of rules was drawn up for use by the individual components.

This paper describes the reasons behind the protocol development and its actual structure. It will also outline the benefits that were gained by its introduction. In particular the following points will be detailed:-

1) The algorithm used to control the data exchange.
2) The code mechanisms used to interpret the data communicated.
3) The command structure
4) The types of command that were implemented.
5) The error handling mechanism.
6) The use of user labels in the component program files.

Although some of the descriptions given in this paper may be specific to the HPDeskManager development, it is intended that the ideas be presented in a way that could be adopted by anyone wishing to introduce such a structured environment into their workplace.

THE REQUIREMENT

HPDeskManager is Hewlett-Packard's Electronic Mail and Transport system and as such represents the hub of the Personal Productivity Center (PPC). As part of the development from HPDeskManager-3 to HPDeskManager-4 it was seen as vital to integrate the individual components of the PPC as closely as possible with HPDeskManager.

These components ranged from products in their own right such as HPWord and HPListKeeper to the many new converters which were being developed to allow the acceptance and translation of many different document types within HPDeskManager. In this environment it was seen immediately that each of the components scheduled for tighter integration would have to perform a series of remarkably similar tasks in order to be called from HPDeskManager as a son process. It was also seen that HPDeskManager would need to repeat a large number of similar operations in order to call each of the processes as its father.

The design was required to be such that HPDeskManager would follow one set procedure (hence one set of code routines) to call any particular component process which then made it necessary for the called program to behave in a set manner. Achieving this requirement would mean that HPDeskManager could follow the same path whether the user required to read, for example, an HPWord document or an HPListKeeper document. It was also necessary for HPDeskManager to pass file names to the processes it had called to specify details such as input and output files and for the called process to return such information as errors.

In the case of some components, it was necessary for the programs to run in a mode where they stopped the execution of HPDeskManager whilst they performed their tasks. Other components were dependent on the fact that HPDeskManager would remain active during execution to read and interpret their converted output.

HPDeskManager should also be able to cope with a system which did not have converters or PPC components that could support conversation with it because they did not contain the necessary intelligence. This could be the case where an old version of a component remained whilst the version of HPDeskManager was upgraded. Any attempt at communication from HPDeskManager to this component would be met with silence, a situation that HPDeskManager should be able to notice.

The obvious solution to the problem was to specify a number of guidelines that could be followed by HPDeskManager and the component products whenever integration was required. This then caused the birth of the protocol.

THE METHOD

COMMUNICATIONS

The first area of the requirement considered was that of the way in which HPDeskManager and its components running as son processes would communicate. As the components were all stand alone programs and had no way of accessing the content of HPDeskManager's Data Base, it was obvious that the files for use by the components would need to be exported into the MPE file system before the required operation could take place. This meant that the component programs would need to be made aware of the Input and Output file names required by HPDeskManager and that HPDeskManager would need to be aware of the results of the components execution so as to know whether to import the file back to its Data Base. Also, as many of the component programs could perform more than one task (e.g. HPWord and HPListKeeper) it would be necessary for HPDeskManager to instruct the son which task to perform.

There were two choices on how to communicate this information. Inter Process Communication Files (Message Files) could be used to transfer the data or the SENDMAIL and RECEIVEMAIL intrinsics. The Message File method of communication was not used as it was seen as a inappropriate for this particular application. HPDeskManager already used a large number of external files and file equates and having to create an extra two files (at least) for each application called was seen as a negative point.

It is true that the SENDMAIL/RECEIVEMAIL method of data transfer is not the fastest available however as the amount of data to be transferred was not large, in fact never to exceed 500 words at any one time, this overhead was seen as acceptable as it provided a tidy implementation requiring only the input and output data files to be created.

THE STANDARD ALGORITHM

In order to stabilize the way in which the Father and Son processes interacted during their communication phase it was necessary to devise a standard procedure which could be followed by both. For this purpose we developed an algorithm which detailed their actions.

A number of considerations needed to be made at this point.

1) Does the father need to create the son or does he already exist.
2) What information does the father need to send to the son.
3) What happens if the son does not understand the father.
4) What happens if the Son terminates abnormally.
5) What if the son encounters an error.

The solution to 1) above is simple, the father is in control of what processes he has created and so will keep track of them. We stipulated that the Son would remain active unless expressly terminated by the father. This was done to avoid the overhead of repeated CREATE intrinsic calls if the user is performing repetitive tasks. It is a simple procedure from the sons point of view, unless the father sends a specific instruction, the son will loop back to a RECEIVEMAIL instruction until either it receives new instructions from the father or the father terminates which automatically kills all of its sons.

The types of information transferred between father and son ranges from input and output file names to execution commands. These are disused in more detail later in this document.

Point 3) above can be considered at different levels. Firstly one can foresee the situation where the father conforms to the protocol and the son does not. In this situation it is important to stop the father ever attempting to call the son. This is catered for by the protocol stating that each son must have a User Label identifying itself as a supporter of the protocol. The second scenario involves two supporters of the protocol, but with incompatible versions. This is handled by the message structure itself.

The son terminating abnormally is a very real problem. To overcome this the concept of using a Job Control Word (JCW) as a flag between the two processes has been introduced. The father sets the JCW to an unacceptable value before transferring control to the son. The son resets this JCW to a suitable value only if it processes the information correctly. This JCW is also used to solve point 5) where it can be set to a specific value if a recoverable error occurs in the sons processing. This can be spotted by the father and appropriate action taken.

Taking these points into account, the following algorithm was arrived at to detail the processing steps for both the father and the son processes.

Father Process	Son Process
Set the communication JCW to 999.	
Check that the son supports the protocol by reading the User label. If not then the son cannot be executed so exit.	
Check the Son process exists and CREATE it if necessary.	
Activate the Son.	
	Son process performs initialization and then suspends on RECEIVEMAIL intrinsic.
Formulate the command string to send to the Son process. Use SENDMAIL intrinsic to transmit.	
Use SUSPEND intrinsic to await activation by the Son.	
	Parse the command string passed and perform the necessary tasks.
	Formulate the command string to return to the father.
	Transfer commands to Father via SENDMAIL intrinsic.
	Set communication JCW to zero for success or > 800 for error and ACTIVATE the father.
	Loop back to suspend on the RECEIVEMAIL intrinsic unless the father has specifically requested termination.
Check communication JCW value and continue.	

The father then reacts in the following way depending on the value of the JCW. If JCW=999 then the Son process has terminated abnormally and the Father has been restarted by MPE. If JCW=0 then the Son has worked. The father should loop on the Pause and RECEIVEMAIL intrinsics until the Sons communication is received. If JCW<>999 and JCW<>0 then an error has occurred during processing and the son will return a textual error message in its command string.

PIPELINING ALGORITHM

As mentioned in the requirement not all Son processes would need to have their fathers idle whilst they were executing. In fact, for some processes such as converting a particular document type so that it could be read, the father must be active. We termed this method of execution **Pipelining** and typically it involves using an IPC file as the output file for the son process.

As can be expected the algorithm required for this method is slightly different from that above. The main difference is that the father process is not suspended by its son and so does not require activation at the process end. This means that the communication between the processes is changed and that both need to be aware of the state of the IPC file (or Pipe) at all times as it is from the status of this that the first indication of change will come. The message interchange is still performed by the SENDMAIL/RECEIVEMAIL pair and the detailed algorithm follows.

Father Process	Son Process
Set the communication JCW to 999.	
Check that the son supports the protocol by reading the User label. If not then the son cannot be executed so exit.	
Check the Son process exists and CREATE it if necessary.	
Activate the Son.	
	Son process performs initialization and then suspends on RECEIVEMAIL intrinsic.
Build an IPC file for communication with Son.	
Formulate the command string to send to the Son process. Use SENDMAIL intrinsic to transmit.	
Loop around using timed FREADs on the IPC file and processing the result.	
	Parse the command string passed and perform the necessary tasks.

Father Process (continued)

Son Process (continued)

If an FWRITE to the pipe returns EOF then shut down as if conversion had completed successfully (Father has requested termination of job).

Formulate the command string to return to the father.

If the pipe has not been opened then open it and write a few blanks to it before closing. This avoids a hang.

If the pipe is still open then close it to inform the father of the son process termination.

Transfer commands to Father via SENDMAIL intrinsic.

Set communication JCW to zero for success or > 800 for error.

Loop back to suspend on the RECEIVEMAIL intrinsic unless the father has specifically requested termination.

Check communication JCW value and continue.

The father then reacts in the following way depending on the value of the JCW. If JCW=999 then the Son process has terminated abnormally and the Father has been restarted by MPE. If JCW=0 then the Son has worked. The father should loop on the Pause and RECEIVEMAIL intrinsics until the Sons communication is received. If JCW<>999 and JCW<>0 then an error has occurred during processing and the son will return a textual error message in its command string.

COMMAND STRUCTURE

Perhaps the facet that makes this implementation of Process to Process communication most different is the assigning of a fixed command structure to the data passed between the processes. This was a relatively simple task but meant that we could open the protocol to a wide variety of users with ease.

Following the decision to use the MPE SENDMAIL intrinsic and the realization that the amount of information to be transferred was not large, an arbitrary limit of 500 words was placed on the size of command buffer that could be passed between programs. This has not so far proved to be a limitation and has the added advantage of reducing the stack size that people need to make provision for in their programs.

Rather than impose a rigid command structure, it was decided to implement one that was self defining. The approach was taken that if the command defined its own length then a variable number could be packed into the 500 words available in the SENDMAIL string. The commands have the following format.

Word 1	Command Number - a signed integer.
Word 2	Command Length - a signed integer. Equals the word count of the entire command. i.e. 2 + number of words in the data.
Word 3 onwards	Data, the format of which is defined by the command number.

As an example of the command structure, the commands that follow are those that would be sent by a father to a son if the father required the son to edit a document. The character / is used to indicate a word boundary.

Father 601/4/PR/OG/602/6/A./01/.0/1 /701/5/FI/LE/X;/502/2/999/2

Son 603/6/A./00/.0/1 /999/2

The following command definitions apply.

502 Edit a file
601 Identity of father
602 Version of father
603 Version of son
701 Name of file to edit
999 End of data

CALLING MECHANISM

As mentioned earlier, each of the son programs required a user label to be inserted in order for the protocol to function correctly. As HPDeskManager was to handle a large number of different data types in the same way (reading, writing etc) it needed to be able to call the relevant son process to perform the task without performing any special procedure. Two aids were introduced to enable this to happen.

Firstly, two files were set up that contained details of the program to run in order to translate one text type into another or edit a particular type of file. The translation file took the form "To translate text type X into text type Y then run program Z" whilst the edit file contained information of the type "To edit a document of type X then run program Y. This enables the father process to treat all son processes in the same way and only to have detailed knowledge about the file type that it is operating on.

As any program that is called may need to have parameters, INFO strings or special entry points, it was necessary to have a way for the father program to find out more about its sons at run time. The most self contained way found to perform this was to build a user label into the son program file. This would have two benefits, firstly, non existence of the label would show the the program did not adhere to the rules set down and so could not be communicated with. Secondly, the label could carry all of the variable information that might be required to run a program at any time.

Again the label required some fixed construction and the definition of it forms part of the protocol. The content can tell the father information such as the program version, which SL to run with and any PARM value that should be used. The user label format is as follows.

Byte	Content
0-5	Label identifier which reads "OPDLAB". This can obviously be any 6 characters depending on the application.
6-15	Version number of the product in the form V.uu.ff.
16-17	Treated as a word. The bits are flags. Bit 16: =1 the program supports the standard protocol. Bit 15:14 The SL to be used when running the program. =0 System Library =1 Account Library (Lib=P) =2 Group Library (Lib=G)
18-25	Textual name of an entrypoint to be used to enter the program. This can be blank.
26-27	Value of PARM to be used when calling the program. May be zero.
28-67	Textual string to be passed to the program as an INFO= string. May be blank.
68-75	Name of the JCW to be used to indicate the success or failure of the called process.

It is worth noting at this point that though this user label in each son process was obligatory for this implementation of the protocol, it need not be so for less complex implementations. We inserted the label in order to ease the father process' task when calling a large number of similar processes. If the father is only calling a few processes, then it is reasonable to expect the father to have enough power to differentiate between the sons it is calling without the requirement for them to have user labels.

THE RULES

This section will not detail each command implemented in the protocol, but will deal with the major groups of commands that have been defined for use. This should give a general overview of the way in which the protocol has been put to use in practice. Each of the groups is referred to by number and the individual commands within that group control associated tasks.

Group	Function
900	Data termination commands. There are only two commands in this group, 999 which is placed at the end of each buffer to signify the buffer end and 901 which instructs the son process to terminate after execution. 901 is not a data termination command but did not fit elsewhere.
800	Error information. These commands are only ever transmitted from the Son to the father. The data component contains a textual error message usually taken from the message catalog of the son process. Certain commands in this group have specific meanings and any error not specifically covered is handled by the general 801 command.
700	File information commands. This commands are used to transfer the Input and Output file details to the son process by means of character strings included in the commands.
600	Program name and Version commands. This group of commands allows the father and son processes to communicate information about their identity. One special pair of commands that we implemented were 606 and 607. Issuing 606 from the father process causes the son to reply with a 607 carrying all of the command numbers that it will understand and attempt to execute. Useful in the case where incremental versions of fathers and sons coexist.
500	Operations to be performed by the son process. Again only issued by the father, these commands control the tasks performed by the son. This is the largest group of commands and allows operations such as "create and edit a new file" to "concatenate files A and B and output them as C".
300	Command types. Only two commands have been defined for this group, 301 and 302. Command 301 signifies that the next command is optional. 302 is returned by the son to signify the result of the optional command. This means that a father can send commands to a son that it is not sure that it can execute.

CONCLUSIONS

From our own internal use of this method for structuring process to process communications, we have found the rules very useful. The protocol is by no means the ultimate answer to all program to program communications problems, however it is a good starting point.

The protocol is now widely used within the Office Productivity Division to govern programmatic communication and is regularly reviewed, enhanced or expanded to meet specific needs. Although informal, it has been accepted by engineers as the starting point for implementations and so it is being proved as a standard which works in practice.

The command structure is flexible and self defining which makes it very easy to use. The SENDMAIL/RECEIVEMAIL intrinsic pairs function adequately to transfer information and we have found the practical application of the rules very simple, one of the advantages being that common code can be used between son processes to handle the communication. The command structure could just as easily be transferred to communication via IPC files. All that would be required would be a small change to the governing algorithm and the replacement of the SENDMAIL/RECEIVEMAIL intrinsics with FREAD and FWRITE calls.

This paper has been an outline of the way in which we at the Office Productivity Division have used the idea of structuring the communications between programs. It is not intended to be a set of strict rules to be adhered to but a series of guidelines that you as customers may find useful the next time that you are communicating between programs.

BIOGRAPHY

Malcolm Sharratt is a software development engineer at Hewlett-Packard's Office Productivity Division. He has 8 years experience of the computing industry and for the last 6 has worked on HP3000's. Malcolm has been with HP for the last three years and has worked on the design and development of HPListKeeper and HPFile/Library.

Connecting HP's Electronic Office to IBM's DISOSS

Mike Shaw
Hewlett Packard
Office Productivity Division, England

Introduction

What this paper talks about

This paper discusses Hewlett Packard's Connection between its electronic office, HP DeskManager (HP Desk for short), and IBM's mailing and library system, DISOSS. It describes:

- The features that the connection offers both the HP Desk user and the DISOSS user.
- Features that make the connection easy to configure and administer.
- Features that allow the administrator to easily trouble-shoot the connection should it go wrong.
- How the connection transmits mail messages from HP Desk to DISOSS.
- How the connection receives mail messages from DISOSS to HP Desk.
- How the connection interacts with DISOSS's library facilities.

Why Hewlett Packard connects to IBM's Electronic Office

Hewlett Packard realizes that it must allow users of its office products to live in an environment that contains more than one Electronic Office vendor. One of the most important Office vendors is IBM.

IBM have two Electronic Office systems that run on their mainframes. These are PROFS and DISOSS.

DISOSS runs under both the large MVS operating system and the smaller DOS/VSE operating system. DISOSS is IBM's strategic Office system, using SNA protocols to communicate.

IBM users are able to connect all of their Office mini-computers (8100, 5520, and System 36) to a mailframe running DISOSS. It therefore provides an integration vehicle for IBM's Office.

PROFS runs under the VM operating system and until recently, only communicated using BiSynchronous datacomms.

Hewlett Packard has products that connect HP Desk to both of these Office systems. They are HP OfficeConnect to DISOSS and HP OfficeConnect to PROFS.

This paper discusses HP OfficeConnect to DISOSS (known as HPOC/DISOSS for short).

The next section highlights the features that HPOC/DISOSS brings to the person who uses, administers, and trouble shoots the connection.

HPOC/DISOSS features

What can HPOC/DISOSS connect to ?

HPOC/DISOSS will connect a network of HP3000s running HP Desk to any IBM mainframe running DISOSS. Not only can mail be sent from HP Desk to DISOSS, but it can also be sent from one HP Desk network to another across DISOSS.

How it the connection appears to the user

In designing HPOC/DISOSS, we wanted HP Desk users to be able to send mail to DISOSS users just like they would to any other HP Desk user. We also wanted DISOSS users to send to HP Desk just like they would to any other DISOSS user. This we achieved using HP Desk's "Foreign Service Connection" as explained in more detail later.

HP Desk users are not only able to exchange mail with DISOSS users, they may also file, search, delete, and retrieve documents from DISOSS's library facility. DISOSS library requests are issued from HP Desk. And the issuing HP Desk system can be anywhere in the HP Desk network.

The results of a DISOSS library request are returned to the requester's InTray once the request has completed. Because the user may well not remember what request they issued, the request details are returned with the results. Thus, a user may look at the item in his InTray and see request and results together.

Administration

All configuration associated with the connection can be done while it is in use.

The configurator was written using VPLUS/3000. This allowed us to make it a "fill in the form" interface, and thus easy to use. We also provided help screens for all of the menus.

Starting and stopping of the datacomm link to the IBM mainframe can be made automatic and is based on powerful start and stop conditions. The link can be started and stopped based on:

- The number of messages waiting to go DISOSS
- The time of day
- Number of messages AND time of day (For example, "Start when more than 3 messages are waiting to go AND when it is after 9:00 am")
- Number of messages OR time of day (For example, "Stop when less than 10 messages are waiting to go OR when it is after 10:00")

Trouble Shooting

We aimed to make the product easy to support. We also tried not to make it like a "black box" with messages going in one side and coming out the other. To achieve these goals, we produced a "manager program". This program acts as a spy hole into the connection software.

From the "manager program" you can:

- Look at the queues of mail messages waiting to go to DISOSS
- Change the order of messages within this queue
- Look at the state of the datacomm link
- Manually override the automatic link control settings (you could force the link to close, for example)
- Look at the log files that HPOC/DISOSS generates while it is running. This may be done when the connection is active or stopped.

The next section looks at how HPOC/DISOSS actually sends messages to DISOSS.

Sending mail to DISOSS

Overview of sending

When HP Desk notes that a message is to go to someone who doesn't live on HP Desk (on DISOSS, for example) it gives the mail message to a program that can handle mail for the non-HP Desk system. In the case of DISOSS mail, HPOC/DISOSS is waiting to send this mail to DISOSS.

To send mail to DISOSS, HPOC/DISOSS must package the mail into an envelope. The envelope is built using the Document Interchange Architecture (DIA) standard. So HPOC/DISOSS takes the mail that HP Desk has given it and converts it to DIA.

HPOC/DISOSS then sends the DIA envelope to DISOSS using IBM's System Network Architecture (SNA) datacomms. The particular brand of SNA that HP Desk uses has two names. It is either called Advanced Program to Program Communication (APPC) or LU6.2.

Figure 1 shows this overview.

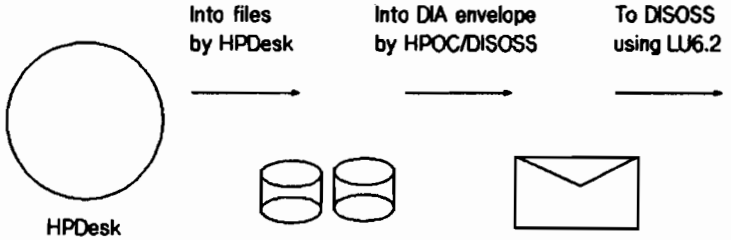


Figure 1: Overview of sending to DISOSS

From HP Desk to HPOC/DISOSS

People within HP Desk have addresses which are made up of their name and a node. The node is 8 characters long. A name might be:

Jackie Lodder / Office

"Jackie Lodder" is the name, and "Office" is the node.

A node can either be an HP Desk node, or it can be a node on a foreign mail system. If the node is a node on a foreign system, HP Desk will take the mail to a computer designated the "gateway computer". When mail is destined for DISOSS, the gateway computer is the one on which HPOC/DISOSS is installed.

On the gateway computer HP Desk will hand the mail to the program that talks to the foreign system. In our case, the foreign system is DISOSS, and the program (or series of programs) is called HPOC/DISOSS.

This interface between non-HP Desk systems and HP Desk is known as the "Foreign Service Connection" or FSC for short.

Lets look at an example of sending mail to DISOSS.

Lets imagine that the HP Desk user, "Jackie Lodder /Office" sends a mail message to someone on DISOSS. The person she is sending to lives on DISOSS, but she knows him as "Richard Madeley" on the node "DISOSS".

Jackie sends her mail message to "Richard Madeley / DISOSS". She does this in exactly the same way as she would if she were sending to another HP Desk user.

The mail message goes through the HP Desk network as any other HP Desk message would.

One of the HP Desk computers in the network is connected to DISOSS. When the mail gets to this computer, HP Desk realizes that the node "DISOSS" is not an HP Desk node.

Its tables tell it that to send mail to anyone on "DISOSS" it must hand the mail over to HPOC/DISOSS.

HPOC/DISOSS is waiting. It picks up the mail message and starts to get it into a form that DISOSS will understand.

Converting the contents parts

HP Desk gives a mail message to a Foreign Service application, like HPOC/DISOSS, by putting the data into a series of files.

The first file is a header file. This file contains the addresses of the people the mail is intended for, the subject of the message, its priority, acknowledgment levels requested, its confidentiality, and its sender's address. The header file is layed out using the "ARPA" standard.

The header file also points to "contents files".

An HP Desk mail message can contain any number of parts. Each part can be of any type. A mail message could contain some graphics, a spread sheet, some text, and a voice item, for example. HP Desk creates one content file for each part of the mail message.

The first thing that HPOC/DISOSS does when it gets hold of the message header file from HP Desk is to squeeze the contents parts into one document. It has to do this because DISOSS will only accept one document per mail message. The document format that DISOSS uses is IBM's Document Content Architecture, or DCA.

The DCA standard does not allow for the inclusion of graphics, voice, image or spread sheets, but HP Desk mail messages may contain such items. This is how HPOC/DISOSS squeezes a multi part HP Desk message into one DCA document:

- If there is more than one content part to the message, then all of the parts are converted to plain ASCII. They are then joined together into one large ASCII file, and the whole lot is converted to DCA.

- If there is a content part that cannot be converted to ASCII, then HPOC/DISOSS will substitute a message for the original content part explaining why loss of information occurred.

There are in fact two types of DCA. There is Final Form Text and Revisable Form Text. Final Form Text cannot be edited - it is intended to be printed only. Revisable Form Text is editable.

Hewlett Packard has converters for both Revisable and Final Form Text. You may select in which flavor of DCA you would like your mail messages to be sent to DISOSS. The selection is done on a "per DISOSS connection" basis.

Lets continue our example.

Jackie's message contained a plain ASCII item and a spread sheet.

Because this mail message has more than one part, HPOC/DISOSS realizes that it will have to join the parts together. To do this, it needs to convert them all to ASCII.

The first part is already ASCII.

Because there is no "Spread sheet to ASCII" converter, HPOC/DISOSS creates a small ASCII file explaining that the spread sheet could not be converted into DCA.

HPOC/DISOSS then joins the parts into one text file. It gives this text file to either the Revisable Form or the Final Form converter to convert it into DCA.

We now have one DCA document.

Mapping the addresses

Before we can get DISOSS to send our document to people in the DISOSS world, we must convert their names from HP Desk format to DISOSS format.

As we mentioned earlier, HP Desk addresses are like normal names with a node name appended to them. DISOSS addresses, on the other hand, have to be an 8 character name followed by an 8 character node name.

The address conversion is done using an "Address Conversion Table". This table is part of the HPOC/DISOSS product. It contains mappings from HP Desk address forms to DISOSS address forms for mail transmission. It can map from DISOSS address form to HP Desk form for mail reception.

Lets go back to our scenario.

Jackie sent to someone with the HP Desk style address of "Richard Madeley / DISOSS".

When the message is exported from the HP Desk world into files, HPOC/DISOSS picks up the header and looks up the destination address of "Richard Madeley /DISOSS" in the HPOC/DISOSS Address Conversion Table. It finds the DISOSS address "RMADY012 IBMHST02". It uses this DISOSS style address to tell DISOSS who the message should go to.

Putting it all in an envelope

Once all of the contents files have been converted to the type of DCA document required and all addresses have been mapped to DISOSS format, everything has an envelope placed around it. The envelope must conform to the Document Interchange Architecture (DIA) standard for DISOSS to accept it.

The envelope contains the DCA document, the subject, the DISOSS addresses of the people the message is going to, the privacy, and the urgency of the message.

In our example, we have a document, and all recipient addresses have been converted from HP Desk format to DISOSS format. Next, HPOC/DISOSS puts all of this information into a DIA envelope.

The message sender

Now that we have a DIA envelope, we have to give this to DISOSS.

The HP3000 uses a version of SNA datacomms that makes it look like a single person word processing work station, the Display Writer.

When a Display Writer user sends a message to DISOSS, he signs onto a user on DISOSS. Thus, for every person who wants to use DISOSS from a Display Writer, there must be a user on DISOSS.

Because the HP3000 looks like a Display Writer, there must be a user on DISOSS for every HP Desk user that wants to exchange mail with DISOSS. We call these users "shadow DISOSS users".

Once the DIA envelope is ready to send, this is what HPOC/DISOSS does:

- It looks up the HP Desk form of the sender's address in its Address Conversion Table. This lookup gives it the name of the "shadow DISOSS user" for the real HP Desk sender.
- HPOC/DISOSS then uses this "shadow user" name as the sender of the DIA envelope.

DISOSS looks in the envelope for the recipient names (now in DISOSS form). DISOSS delivers a copy of the DCA document to each one.

When the DISOSS recipient looks at the message it appears to have came from a DISOSS user. The address the recipient sees is that of the shadow user associated with the real HP Desk sender.

Let's see how that works with our example.

"Jackie Lodder / Office" is the HP Desk sender of our mail message. When HPOC/DISOSS looks up this address in its Address Conversion Table, it gets the DISOSS address "JLODDE01 DISHST01".

HPOC/DISOSS then sends the DIA envelope it built to DISOSS. It tells DISOSS that the sender address to use is "JLODDE01 DISHST01".

The message has one recipient - "RMADY012 IBMHST02". DISOSS therefore sends the DCA document to this person.

When "RMADY012 IBMHST02" looks at the mail message, he sees that it came from the shadow DISOSS user "JLODDE01 DISHST01".

Now let's look at how mail gets from DISOSS to HP Desk.

Receiving mail from DISOSS

Overview of receiving

Here's a brief overview of how HPOC/DISOSS gets mail from DISOSS to the recipient's InTray on HP Desk

HPOC/DISOSS asks the DISOSS mainframe if there is any mail for anyone on HP Desk. If there is mail, DISOSS sends it to HPOC/DISOSS.

HPOC/DISOSS accepts the mail. It takes the DCA document out of the DIA envelope and puts it into a file. It also converts all of the addresses from their DISOSS form to their HP Desk form.

HPOC/DISOSS then creates an ARPA header just like the one that HP Desk gave to HPOC/DISOSS when mail was going to DISOSS. Once the header is created, HPOC/DISOSS tells HP Desk to pick up the mail message.

HP Desk takes the header and the DCA document and imports them into an HP Desk mail message. This message then travels on to its intended recipient using the usual HP Desk transport mechanisms.

This overview of receiving is shown in figure 2.

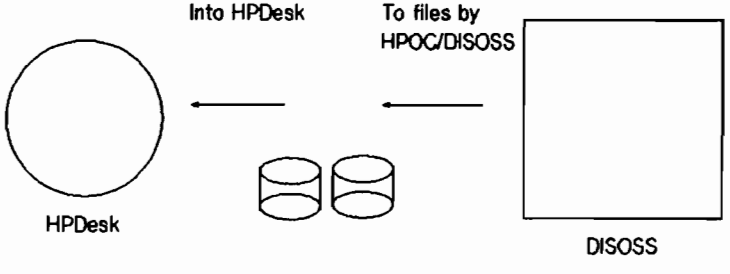


Figure 2: Overview of receiving from DISOSS

Asking DISOSS for mail

The first thing HPOC/DISOSS has to do is to ask DISOSS for mail.

We mentioned earlier that HPOC/DISOSS makes an HP3000 look like a single person work station, the Display Writer.

When a Display Writer user wants to read his mail, he has to ask DISOSS "do I have any mail". DISOSS looks in the "InTray" that it keeps for this user on DISOSS. If there is mail, DISOSS sends the mail to the Display Writer.

HPOC/DISOSS must therefore do the same thing. From its Address Conversion Table, HPOC/DISOSS forms a list of people on HP Desk who wish to exchange mail with DISOSS. For each HP Desk user on the list, it has the HP Desk name and the DISOSS shadow user name.

HPOC/DISOSS goes down this list, and for each person on the list, it asks DISOSS "Does this shadow user have any mail ?". If there is mail, DISOSS will send it. Figure 3 shows this.

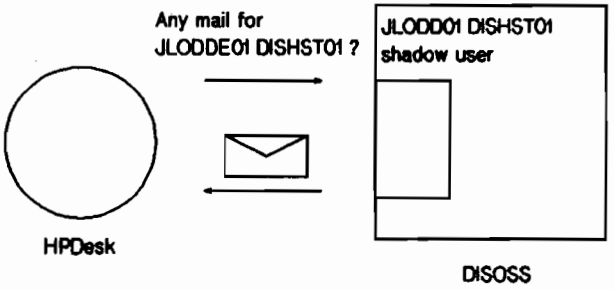


Figure 3: Asking DISOSS for mail

HPOC/DISOSS keeps cycling around this "polling" list. The frequency with which it cycles is configurable.

Let's see how this works with an example.

Our HP Desk user "Jackie Lodder / Office" has a shadow user on DISOSS with the address "JLODDE01 DISHST01". This shadow user address will be one of the users on HPOC/DISOSS's polling list.

Let us imagine that Jackie has some mail. Let us further imagine that the mail is from the real DISOSS user, "RMADY012 IBMHST02".

When HPOC/DISOSS asks "Is the any mail for JLODDE01 DISHST01 ?", DISOSS will reply by sending a DIA envelope with the mail in it.

If there were two mail messages for Jackie, DISOSS would just send two envelopes.

High speed mail collection

We can speed up this polling for mail, but at a cost.

On DISOSS we can define a special shadow user to be a delegate of all the other shadow users. Now, rather than having to ask for mail for each individual shadow user, we ask only for mail for the delegate and all its principals. Thus we can collect mail for a whole HP Desk network with just one poll. Figure 4 shows this.

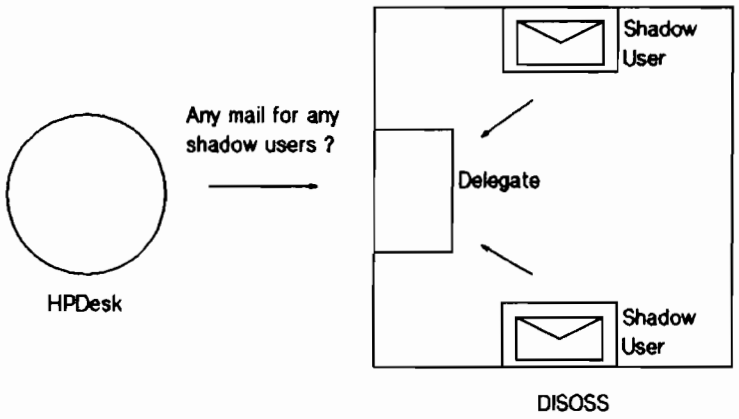


Figure 4: High speed collection

What is the catch ? Well, delegate polling can't pick up private mail.

To get the advantages of both individual polling and delegate polling we have come up with a hybrid scheme. HPOC/DISOSS does one "delegate poll" for every four "individual polls". This way, all non private mail is delivered quickly, but private mail is also collected.

Breaking up the DIA

We now have one or more DIA envelopes on the HP3000. HPOC/DISOSS treats the envelopes in reverse to the way it did when sending:

- It extracts the DCA document from the envelope. It puts it into a file. The "type" of the file indicates which sort of DCA it is - Revisable or Final.
- It takes all of the addresses in the envelope (both sender address and recipient addresses) and converts them into HP Desk type addresses. It uses its Address Conversion Table to do this.
- It looks through the envelope for other things to put into the header file. These include the subject, the privacy of the message, and the urgency of the message.
- It creates a header file. The format of this header file is exactly the same as that of the file that HP Desk gives to HPOC/DISOSS when a message goes to DISOSS.

In our example:

The sender's address is "RMADY012 IBMHST02" and the intended recipient is "JLODDE01 DISHST01".

After HPOC/DISOSS has looked these two addresses up, the header file says that the mail message was from "Richard Madeley / DISOSS" and is to go to "Jackie Lodder / Office".

Richard sent a Revisable Form DCA document. This is exported from the DIA envelope and put into a file of type Revisable Form DCA.

From files to HP Desk

Once HPOC/DISOSS has put the DCA document into a file and has created a header file, it tells HP Desk to pick up the files and create a mail message from them.

HP Desk then turns the header file into a normal HP Desk mail message using the DCA file as the message's content part. It sends the mail message to the intended recipient as indicated in the header file.

Back to our example:

HP Desk picks up the header and the DCA document and makes them into a normal HP Desk mail message. The intended recipient is "Jackie Lodder / Office". HP Desk delivers this to her InTray exactly like it would any other HP Desk message.

When Jackie reads her message, it says that it came from a "Richard Madeley / DISOSS". All trace of the DISOSS name, "RMADY012 IBMHST02" has gone. To Jackie the message looks like any other HP Desk message she might receive.

Although the content is DCA, she is able to view this too. This is because there are converters to convert from both RFT and FFT DCA to Hewlett Packard's internal document formats. These are called automatically by HP Desk - Jackie is unaware that the conversion is taking place.

Using DISOSS library functions

Introduction

DISOSS's library functions are accessed from within HP Desk. Any user in the HP Desk network may issue a library request.

The following DISOSS library functions are available to HP Desk users:

- They may file a DCA document on DISOSS. The document is filed with a document profile. This profile describes the author, subject, date filed, and who can access the document. The profile may also contain up to 5 keywords. This profile information is used by DISOSS when a search is requested.

- They may file a reference to a physical object. This is just the same as filing a document but no electronic document is filed. Instead, the profile refers to something physical like a mechanical part or a paper book.

- They may issue a search of the DISOSS library. When specifying the search criteria, the user may choose to search on author, date filed, keywords, and subject. The "space" in which the search takes place is specified at search time. The "space" can be:

  - "Documents filed on DISOSS by this HP Desk user" or
  - "Documents to which this HP Desk user has access".

- They may retrieve documents from DISOSS's library. The result of a DISOSS search is a list of document profiles that meet the search criteria. This list is returned to the requester's InTray. From there, he may select to retrieve one or more documents from this list. The documents he selects will be returned to his InTray in a "library response" mail message.

  There is another way of retrieving but this is more awkward to use. When an item is filed on DISOSS, a unique ID is associated with the item. This ID is returned to the HP Desk requester via a "library response" to his InTray. If he remembers this ID information, by filing it in his personal HP Desk filing cabinet for example, he may request to retrieve the item by specifying the ID.

- They may delete documents from the DISOSS library. As with retrieving, the user may elect to delete documents by selecting the document's profile from a search result list.

  Items may also be deleted using the item's DISOSS ID.

What happens when you send off a library request ?

Once a user has formulated a DISOSS library request from within HP Desk, the request is automatically sent to a special HP Desk node. This node is a foreign node. And the DISOSS library request is in fact a normal mail message, the first part of which is a special item specifying the "library request" details. Once the request has been created by the HP Desk user, he is free to get on with something else. When the library request has finished, the request and its response are sent to his InTray.

When the "library request" mail message gets to the computer with HPOC/DISOSS on it, HP Desk exports the message into files just like it did with mail messages destined for DISOSS. If the request is a "file", there will be 3 files : the header, the filing request details, and the document to file. If the request is a "search" there will be 2 files : the header and the search request details.

HPOC/DISOSS is waiting for library requests to pop out of HP Desk. It takes the library request from HP Desk and transforms it into a format that DISOSS will understand. The format that DISOSS uses for library requests is also DIA.

HPOC/DISOSS sends the DIA and waits for DISOSS to reply.

What does HPOC/DISOSS do with DISOSS's reply ?

DISOSS sends its reply to the library request immediately.

If the request was a "file", the reply will contain the item's DISOSS ID. If the request was a "search", the reply will be a list of document profiles that matched the search criteria. If the request was a "retrieve", the reply will be the document requested.

The reply is in a DIA envelope. HPOC/DISOSS takes the reply out of its envelope and creates a "library response" mail message to be sent back to the HP Desk requester.

The returned library response contains not only the request results but also the request details. Thus, when the library response gets back to the requesters' InTray, he is able to read the request he sent out and the response that DISOSS made to that request. Figure 5 shows this.

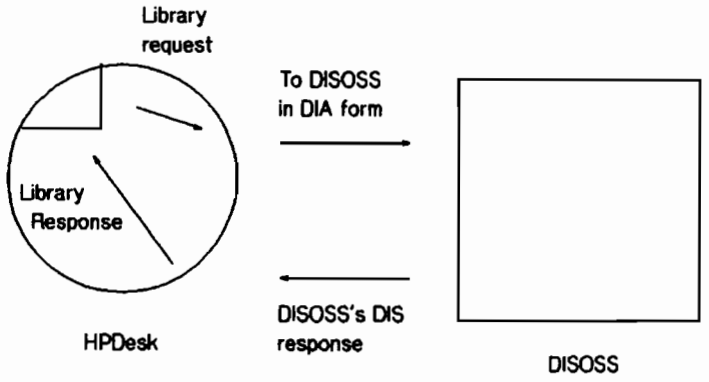


Figure 5: The path of a library request

Let's see how Jackie Lodder might use DISOSS's Library.

Jackie wants to file a Revisable Form document on DISOSS. From HP Desk she specifies that she would like this document filed with the keywords "Example", and "Sales Figures". The subject, author, and create date that the item has within HP Desk are put in the profile for her automatically.

In Jackie's HP Desk network, DISOSS library requests go to an HP Desk node called "LIBRRY"

HP Desk delivers the "library request" mail message to the node "LIBRRY" by exporting it into 3 files.

HPOC/DISOSS picks up the files, and from the "Library request file" it creates a 'file' request in a DIA envelope

HPOC/DISOSS sends the envelope to DISOSS. DISOSS files the document as Jackie requested and sends back a DIA envelope containing the item's DISOSS ID.

HPOC/DISOSS takes this ID from the DIA and creates a "library response" mail message with the original request details and the response ID. It addresses this library response to "Jackie Lodder / Office" and tells HP Desk to pick it up.

The library response is delivered to Jackie's InTray. When she reads it, she sees the original filing request and the ID that DISOSS gave in response.

She files the library response in her personal HP Desk filing cabinet so that she can access the ID should she need to.

Conclusion

This paper talked about Hewlett Packard's connection between its Electronic Office, HP Desk, and one of IBM's strategic Electronic Office product, DISOSS.

It told how the connection is "tranparent" to both DISOSS and HP Desk users.

It discussed how HP Desk's interface to foreign mailing systems was used to impliment a connection to DISOSS's mailing and library services.

MPE Security : In Perspective

Thomas Shem
Hewlett-Packard Company
19447 Pruneridge Avenue
Cupertino CA, 95014

Security on the Multiprogramming Executive (MPE) operating system is often overlooked due to MPE's desire to be an easy to use interface to the HP3000. This usability provides the HP3000 with a great deal of latitude since it allows MPE to operate in a wide variety of environments. On the other hand, ease of use is often in direct conflict with system security since security seeks to limit access to system resources. This conflict has often led users to conclude that MPE has no security, but upon closer inspection this is shown to be untrue.

Our discussion will delve into defining computer security as it pertains to the MPE operating system. To do this we will need to explain some of the basic concepts inherent to secure operating systems. Using these concepts as a base, we will show how they relate to the MPE operating system.

With this basic understanding we will shift our focus to understanding the security provisions which are available in the MPE operating system and how they might be strengthened. Next, we will look at recent enhancements that have been made to the basic operating system. These enhancements do much to increase the security provisions of the system. Lastly, we will introduce some of the improvements which are inherent to the new MPE XL operating system.

Secure Operating Systems : An Introduction

To gain an understanding of computer security, we need to first look at some of the basic theory and concepts inherent to secure operating systems. These concepts provide strategies for defining secure environments and for defining relationships for accessing data or resources.

We begin our discussion by examining basic resource access concepts which every user must abide by in order to gain access to a system resource. A brief description about the concepts of subjects and objects will follow. Next, our discussion will quickly cover protection domains. Finally, we will discuss how security is achieved via the interaction between the protection domain and the system resources, i.e. the protection mechanisms.

Access Concepts:

By generalizing much of the theory and methodology from operating systems security, we can derive three basic access processes. They are identification, authentication, and authorization. The purpose of these three processes becomes clear when we examine them in the context of the sequence of events that occurs as a user logs on, i.e. the logon process.

A user attempting to logon onto a computer must present some type of logon string. This string can be thought of as the user's identification. The characteristic of this identifier is that it is known to both the user and the system.

After the user has input his logon string, the system asks for a password. The process of verifying the user's ID and the user's password is the authentication process. The system first notes that the logon string is valid, then it verifies that the password belongs to that logon string.

The authorization process is more subtle in that once the system has verified that the user is a valid user, the system gives that user access to a set of resources. The authorization process is performed every time the user attempts to access system resources. The system either allows or denies the resource to the user based upon the rights that are associated to the user.

Subjects and Objects:

The concept of subjects and objects is essential to the understanding of operating system security. Basically subjects manipulate data from objects.

Subjects can be thought of as the active entities on a system, the users or processes. They manipulate data or request functions from the system.

Objects can be thought of as the passive entities on a system, the files, system tables, or any repository of information.

An example of the subjects to objects concept is a user reading a data file. The user would be the subject since he is requesting data. The file would be the object, the repository of information.

Protection Domains:

With the multitude of computer systems available on the market, the characteristic which most defines their individuality is their protection domain. The protection domain is the collection of all possible protection environments which may exist on any system at any one time. A protection environment can be thought of as the restrictions and the privileges that determine the relationships between subjects and objects. These restrictions and privileges mediate access to system resources.

As an example, suppose that our system has two environments, one for a ordinary user and the other for a superuser. The superuser would be able to perform any function on the machine such as configure new users, modify system tables, halt the machine, etc. The user environment would be restricted from performing any of these functions and would be restricted to potentially harmless functions. From this example we can see that the protection domain of this system would consist of the superuser environment, the user environment being a subset of this. More elaborate examples would provide for many classes of users, each class being restricted to a subset of the overall protection domain.

Protection Mechanisms:

The relationship established on a system between the protection domain and the subjects and objects describes the system security. That relationship is enforced via various protection mechanisms. The mechanism which determines when a subject may have access to an object uses the restrictions associated with the subject's protection environment. Some examples of these protection mechanisms are protection bits, access control lists, and labeling.

Protection bits refer to indicators which allow a subject to access a particular object in a particular manner. An examples of protection bits are capability lists. A simple implementation of

protection bits associates a flag with the ability for a user to read, write or execute a particular file.

Access control lists are very simple in nature. This protection mechanism refers to a list being associated with a particular object. The list describes users which may access the object and how those users may access it.

Suppose we have a file called "TEST". It has an access list which contains two entries. The first entry describes the superuser's access rights to the file: read, write and execute. The second entry describes the ordinary user's access rights to the file: read. If the ordinary user were to attempt to execute the "TEST" file, he would be denied the execute right. The protection mechanism would not allow the user to execute the file because the access control list did not contain the execute right in the ordinary user's entry.

A protection mechanism inherent to very secure systems is labeling. Labeling is a method whereby every subject and object is assigned a security level. When a subject attempts to access an object, the system compares security levels of each. On simple systems, the subject would have read and execute access to the object only if the security level of the object was less than or equal to that of the subject. On more complex systems, the levels would be evaluated against each other based upon a more complex set of rules.

Our brief investigation of the concepts associated with a secure system shows that they are far from simple. Users of a system are placed into a protection environment, a subset of the entire systems protection domain. These protection environments define a set of access rights for that user.

Within a system, subjects are constantly attempting to access objects. The relationship between the subjects and objects enforce a systems security policy. The enforcement of that relationship is achieved via protection mechanisms. Protection mechanisms determine whether a subject may access an object.

MPE Security : An Introduction

Now that we have an insight into some of the basic theories and concepts of operating systems security we will see how they relate to the basic MPE operating system. As before our discussion will begin with basic resource access concepts identified in the logon process. We will follow with a discussion of the protection environments which make up the protection domain of MPE. Finally, the protection mechanisms used in MPE will be examined.

Access Concepts:

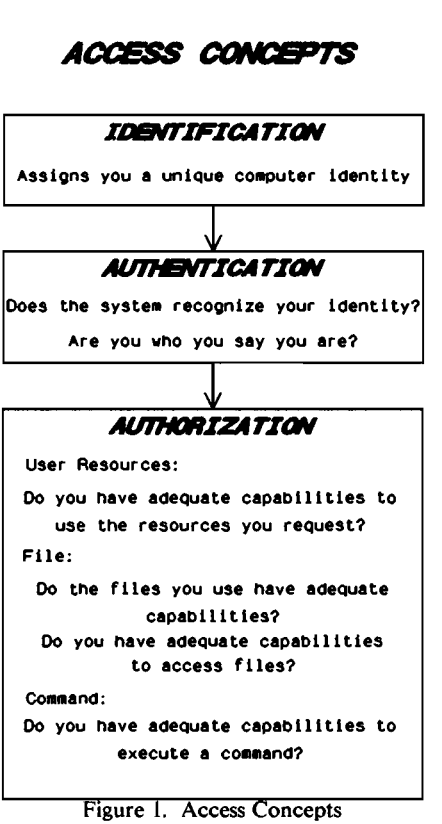


Figure 1. Access Concepts

An examination of the MPE logon process shows that a user is identified by a combination of an 8 character user identifier and an 8 character account identifier. In order for a user to logon to an HP3000 the user must present his identifier to MPE in the HELLO or JOB command.

The authentication process begins when the HELLO or JOB command is recognized by the operating system. MPE determines if it knows who the user is by inspecting the logon identifier. The authentication process also verifies that the logon identifier belongs to the user by means of a user, account, and group password. MPE reasons that if the logon ID belongs to the user, the user knows the passwords.

Now that the user has successfully gained access to the system the constant process of authorization begins. During the life of the user's job or session, various checks are constantly being done. Whenever a user attempts to access any system resource, checks are made to determine whether or not that user is authorized to use the resource.

Protection Domains:

The protection domain on MPE derives its existence from the system architecture. MPE supports two environments, a privileged mode and a user mode. In the privileged environment,

users may gain unrestricted access to every resource on the HP3000. The user environment provides degrees of resource restrictions which are used to define other environments.

User mode in MPE is divided into several classes. These classes define system roles which are characterized by their access rights to system resources. Several classes, system manager, system supervisor, and account manager, are special in that they are allowed access to functions which maintain the working environment on the HP3000.

The System Manager environment allows users to perform all tasks associated with maintaining system integrity and system security. In his environment he has access to every usable system resource.

The System Supervisor environment is a subset of the system manager's environment. He is allowed access to those functions necessary for the normal day to day operations of the system.

The Account Manager environment is also a subset of the system manager's environment. His environment deals with the upkeep and security of a single account on the HP3000.

Protection Mechanisms:

Table 1. Capabilities

Capability	Abbrev-iation	Acct	Group	User
System Manager	SM	X		X
System Supervisor	OP	X		X
Account Manager	AM	X		X
Account Librarian	AL	X		X
Batch Access	BA	X	X	X
Use Communications Software	CS	X		X
Diagnostician Attribute	DI	X		X
Extra Data Segments	DS	X	X	X
Group Librarian	GL	X		X
Interactive Access	IA	X	X	X
Multiple RIN	MR	X	X	X
Network Administrator	NA	X		X
Node Manager	NM	X		X
Use Nonsharable Devices	ND	X		X
Use Private Volumes	UV	X		X
Privileged Mode	PM	X	X	X
Process Handling	PH	X	X	X
Programmatic Sessions	PS	X		X
Save User Files Permanently	SF	X		X
Use User Logging Facility	LG	X		X
Create Volume Sets	CV	X		X

A user's environment is defined by his capability list, a protection mechanism. The capabilities serve two purposes; to define classes of users, each having certain system privileges and to define user rights. The capabilities are associated with users, groups, and accounts (see table 1). The capabilities associated with the users and accounts make up the majority of a user's system resource protection environment.

The capability protection mechanism plays a large part in providing a secure environment in the MPE operating system. We shall now examine the other protection mechanism which exist within that environment which aid in controlling access to MPE commands and files.

Command Access:

MPE provides its users with a large set of commands which allow them a wide variety of functions. These functions range from shutting the system down, to creating a new user, to showing the current time. Obviously, the functions that a user attempts to perform should be consistent to the rights and privileges that he has been assigned. In this regard, a subset of MPE commands are restricted to those users who operate in each of the aforementioned special environments. The following three tables illustrate these special commands.

Table 2. System Manager Commands

:ALTACCT	:NEWACCT	:PURGEACCT	:RESETACCT

Table 3. Account Manager Commands

:ABORTJOB	:LISTACCT	:NEWGROUP	:PURGEUSER
:ALTGROUP	:LISTGROUP	:NEWUSER	
:ALTUSER	:LISTUSER	:PURGEGROUP	

Table 4. System Supervisor Commands

:ALLOCATE	:DEALLOCATE	:RESUMELOG	:SWITCHLOG
:ALTSPOOLFILE	:FULLBACKUP	:SHOWLOG	:SYSDUMP
:CACHECONTROL	:JOBPRI	:SHOWQ	:TUNE
:CHANGELOG	:LOG	:STARTCACHE	:VINIT
:CONSOLE	:PARTBACKUP	:STOPCACHE	

The NEWACCT command is an example of one of these commands which may only be used in the system manager environment. Since it is the system manager's responsibility to maintain system integrity and system security, the responsibility of allocating a new account and account manager belongs to him.

Creation of a new user via the NEWUSER command is an example of a command which falls into the realm of the account manager environment. The responsibility of maintaining the account makes the AM capability requirement necessary for this command; however, the responsibility does not solely rest with the account manager. The ultimate responsibility of maintaining the system integrity belongs with the system manager; therefore, the SM environment is also allowed to execute the NEWUSER command.

From the previous examples we have seen that the protection mechanism to allow access to MPE commands relies on the capabilities assigned to a user. Generally, this mechanism is sufficient to protect access to MPE commands; however, there exists a set of commands which require stricter measures of protection (see table 5). These commands deal with MPE system resource availability. Examples are ABORTIO, UP, DOWN, GIVE, TAKE. The mechanism used to protect these commands restricts them to a specific logical device, the Console. Only in this manner can the system resource availability be adequately controlled.

Table 5. Console Commands

=/:ABORTIO	:FOREIGN	:OPENQ	:STOPSPOOL
=ABORTJOB	:GIVE	:OUTFENCE	:STREAMS
:ACCEPT	:HEADOFF	=RECALL	:SUSPENDSPOOL
:ALLOW	:HEADON	:REFUSE	:SWITCHLOG
:ALTJOB	:JOBFENCE	=/:REPLY	:TAKE
:BREAKJOB	:JOBSECURITY	:RESUMEJOB	:UP
:DELETESPOOLFILE	:LDISMOUNT	:RESUMESPOOL	:VMOUNT
:DISALLOW	:LIMIT	:SHOWCOM	:WARN
:DISCRPS	:LMOUNT	=SHUTDOWN	:WELCOME
:DOWN	=LOGOFF	:SHUTQ	
:DOWNLOAD	=LOGON	:STARTSPOOL	

The protection mechanism used to restrict MPE command access is based heavily upon the access rights of that user (the protection environment). When a user (the subject) attempts to gain access to a resource (the object) by executing a command, the command interpreter verifies that the user has those capabilities necessary to execute the command (the protection mechanism).

File Access:

System functionality is not the only resource which needs protection; files must also be protected. The file protection mechanism is known as File Access Attributes. File Access Attributes are located in the file label, the group and the account. All three file access attributes when taken together determine who can access an individual file.

Table 6. File Access Modes

Mode	Code	Meaning
READ	R	Allows users to read files.
LOCK	L	Permits a user to prevent concurrent access to a file. Specifically, it permits the use of the FLOCK and FUNLOCK intrinsics, and the exclusive-access option of the FOPEN intrinsic.
APPEND	A	Allows users to add information and disc extents to files, but prohibits them from altering or deleting information already written. This access mode implicitly allows the LOCK (L) access mode described above.
WRITE	W	Allows users general writing access, permitting them to add, delete, or change any information in files. This includes removing entire files from the system with the :PURGE command. WRITE (W) access also implicitly allows the LOCK (L) and append (A) access modes described previously.
SAVE	S	Allows users to declare files within a group as permanent, and to rename such files. This includes the ability to create new permanent files with the :BUILD command.
EXECUTE	X	Allows users to execute program files, with the :RUN command or the CREATE and CREATEPROCESS intrinsics.

The file access attributes specify which user type (see table 6) may have access to each of the file access modes; read, lock, append, write, save, and execute. The user type specifies a grouping of valid users under a specified account (see table 7).

As an example, if a file was only to be accessed by its creator it would have a file access attribute of (R,L,A,W,S,X:CR). The attribute permits read, lock, append, write, save and execute modes to the user who created the file. If a file was assigned a file access attribute of (R:ANY; W,L,S:GU; X:AC), read access would be granted to all users; write, lock and save access would be granted to group users; and execute access would be granted to account members.

Table 7. User Types

User Type	Code	Meaning	Acct	Group	Files
Any User	ANY	Any user defined in the system. This includes all catagories defined below.	X	X	X
Account Member	AC	Any user authorized access to the system under this account. This includes AL, GU, and CR users under this account.	X	X	X
Account Librarian	AL	User with Account Librarian capability, who can manage certain files within the account which may include more than one group.		X	X
Group User	GU	Any user allowed to access this group as the log on or home group, including all GL users applicable to this group.		X	X
Group Librarian	GL	User with Group Librarian capability, who can manage certain files within a home group only		X	X
Creating User	CR	The user who created this file.			X

The file access attributes are assigned at the account level, group level, and the file level. This hierarchical level attribute tree complements the MPE directory structure where files are contained in groups which are linked to accounts. A specific account may contain several groups, each being assigned a different file access attribute.

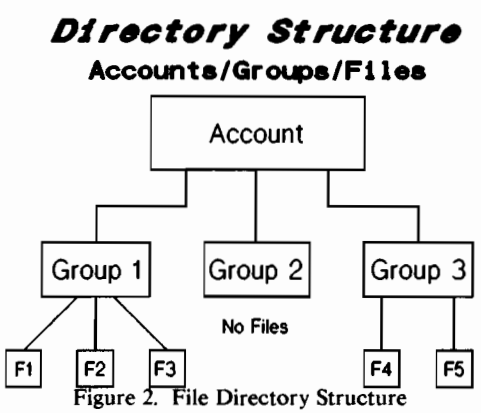


Figure 2. File Directory Structure

The three levels of file access attributes control file access by only allowing certain types of users to access a file.

The account level file access attributes controls access of all files located within the groups under the account. It can be set, for each of the five modes, to allow only account members (AC) file access or to allow all users (ANY) file access.

The group level file access attributes controls access to all files stored within the group. The user types which are controlled at the group level are all users (ANY), account member (AC), group user (GU), account librarian (AL), or group librarian (GL).

The file level file attributes many be set to restrict access to the specific file in as unsecured a manner as the creator of the file desires; however, when a user tries to access the file, the system will check each of the account, group, and file level file access attributes to determine the file's accessibility.

The hierarchical access structure while complex in nature allows a great deal of freedom to HP3000 users. The file access attributes at the account and group level serve only to screen out possible users of the files. The file level file access attribute defines which portion of the screened users may have access to the file.

The protection mechanism used to protect files is initially based upon the protection environment under which the user is operating (the account and group file access attributes). This is further complicated because the file has many modes of access. The user (the subject) attempts to access a file (the object) in a particular mode. The file access attributes from the account, group and file label for the particular mode of access are checked to determine whether the user can access the file (the protection mechanism).

Our brief discussion of some of the basic concepts of secure operating systems has given us an insight into the nature of security and how it relates to the HP3000. We shall now turn our discussion to MPE's security features.

MPE Security Features

A method commonly used to evaluate operating systems security examines the security features available to a user. An examination of each of MPE's security features may give a better perspective as to the security of the system.

Identification Features:

As has been stated in a previous section, a user logs onto MPE via his user identification. On MPE that identification consists of a USER ID and an ACCOUNT ID. Additionally, the user identification can be expanded to also include a GROUP ID.

Authentication Features:

A commonly used method to authenticate valid users on a computer system involves the use of passwords. Passwords on MPE consist of a string of up to eight characters. Passwords may be assigned to the USER, ACCOUNT, and GROUP IDs.

During the logon process, the user is given three chances to correctly enter each of his passwords. During this entry period any passwords that a user might enter are suppressed from printing on the terminal. If the user does not enter the correct password within those three tries, he is denied access to the system. Additionally, the console is notified about the aborted attempt.

Authorization Features:

As we have seen before, each user can be assigned a limited set of capabilities. This capability set is how MPE limits users access to system functionality, i.e. MPE commands. This capability set may also be used to restrict users to certain modes of operation (i.e. batch, interactive, network) and to allow users access to system resources (i.e. private volumes, spooler, plotters).

File Protection Features:

Access to files on MPE can be controlled to a specific access type (i.e. read, write, append, execute and lock). Additionally, access to files can be limited to a subset of users (i.e. any user, account members, group users, account librarians, group librarians, or creators). MPE also provides for the ability to place a password, "lockword", on a file, so that anyone trying to access a file must know its lockword.

Files are further protected via the use of date stamping. MPE saves the date of the file creation, date the file was last modified and date the file was last accessed. Security conscious users can then determine if a file was tampered with.

System Architecture Protection Features:

Often features provided by the system architecture are overlooked. These features are extremely important since they describe the make-up of the machine.

MPE's system architecture protection features include bounds checking implemented in the hardware. This prevents the possibility of normal users corrupting vital information.

Each process on the system is assigned a private, hardware protected data storage area known as a stack. Each stack is kept separated from other stacks. Data from one process can be kept from interacting with another.

MPE also supports two states of operation within the system, a user and privileged mode.

Audit Trail Features:

A feature important to systems security is the logging of important events. These events deal with access to system resources. Events which are logged on MPE include job initiations, job terminations, process terminations, file closes, system shutdown, power fails, line disconnection, line closes and I/O errors. Other important events issue messages to the system console. These console messages can also be logged. Additionally, MPE also provides a user logging facility so that user defined events can be logged at the application level.

Strengthening System Security

The security features provided by MPE are not the only features which can be used to control resource access. Many of the functional features provided by the operating system can be used to support system security.

User Defined Commands:

The user defined command (UDC) facility is a powerful system feature which allows a user to redirect user commands to user defined functions. UDCs can strengthen MPE security features by allowing a system manager or account manager to capture and redirect any MPE command. This capturing of commands increases authorization security by further restricting a users command set.

UDCs can also be used to restrict a user to a single application program. When the user logs onto the system he can be forced into an application environment and away from MPE (LOGON UDC).

The mechanisms provided in the UDC facility along with those of the file system can protect the user defined command from tampering. The UDC facility provides options to prevent the break key from functioning (OPTION NOBREAK/BREAK) and to prevent a user from seeing the defined command (OPTION NOLIST/LIST, OPTION NOHELP/HELP).

The System Catalog:

The MPE operating system error message facility was designed to help the user by issuing informative error messages if he makes a mistake. An example of this occurs during the logon process. If a user types in an invalid identifier an informative error message will tell the user which part of the identifier is wrong, thus giving the user another chance to get it right. If the user is attempting to break into the system these messages will help him to accomplish the deed.

These informative error messages reside in CATALOG.PUB.SYS and can be changed to a less informative message such as "INVALID LOGON IDENTIFIER". Once the desired messages are changed the use of MAKECAT.PUB.SYS will replace the system error message catalog.

Hardcopy Console:

The system console is the recipient of many status messages which are issued by MPE. These messages can prove to be valuable when trying to analyze system accesses. A hardcopy terminal

or a logging terminal will insure that the console messages are not lost as they are on a CRT terminal.

Enhancing MPE Security

The MPE operating system when first introduced provided an excellent set of security features which suited the scope for which the HP3000 was designed. Since its inception, the HP3000 and its current operating system, MPE V, has evolved to fulfill a wide range of applications and environments.

As part of this evolution, a new set of security features have been developed which enhance those mechanisms already present in MPE. The new security features address four major areas; logon, password maintenance, access to the command interpreter, and system logging.

New Logon Features:

System security starts when a user begins to use a computer terminal. A new layer of security has been added to MPE via a terminal password. Specific logical devices may be assigned an eight character password which a user must enter before access to MPE's logon prompt is given.

A common method used to discover passwords involves recursively attempting all possible combinations of a possible password. A recent enhancement has been added to prevent such an occurrence.

The maximum attempts that a user may unsuccessfully attempt to log onto a terminal may be configured into the security system. If a user exceeds that amount, the logical device which he is using will become unavailable. The system manager may configure the system to re-enable the device automatically after a specified period of time if that is desired.

As was mentioned earlier ease of use is often in direct conflict to system security. Three recent enhancements will allow a system manager to place security concerns above those of user friendliness; the first involves the logon interface while the others involve the logon process.

As was mentioned previously one method which can be used to strengthen system security is to modify the system catalog by replacing the helpful logon messages with less helpful ones. A system manager no longer has to modify the catalog to accomplish this. The system manager may elect to use the new feature to prevent helpful logon messages from being issued.

Embedding passwords in both job stream files and user logon strings are not secure practices since they place sensitive passwords in plain text for bypassers to see. The system manager may now cause embedded passwords to be prompted for in both the files and logon strings. For job files, embedded passwords will not be necessary because the system will prompt for them. For logon strings, embedded passwords will be ignored forcing users to respond to system prompts.

New Password Maintenance Features:

We have seen that passwords on MPE play a major role in protecting the HP3000 from unauthorized intrusion. Passwords should therefore be changed periodically to decrease the possibility of their discovery. Recent enhancements have been made to MPE to facilitate the password protection mechanism.

Users have been given control over their own user password. A new command, PASSWORD, has been provided so that all users may readily change their USER password.

In order to enforce the need for secure passwords, the system manager may cause all USER passwords to be invalidated. Users will then be forced to change their USER passwords periodically. The account manager will also have the ability to selectively invalidate USER passwords within his account.

Causing passwords to be changed periodically is of little use if users delete them or make short easily guessed passwords. Two enhancements provide mechanisms to enforce the password facility. USER passwords can be required by the system to prevent null passwords from occurring on the system. Also, the system manager may configure a minimum allowable length for all passwords on his system. This is especially important since longer passwords are more difficult to guess.

Lastly, an enhancement which should be invisible to most users will cause all passwords to be stored on the system in encrypted form.

More Secured Command Interpreter:

Several of the recent security enhancements deal with securing access to the command interpreter (CI). This is important since the command interpreter is the users' highway to system resources. The first is designed to protect the command interpreter from unattended terminal sessions while the others are designed to restrict command access.

Each time that a user walks away from his terminal without logging off the system represents a potential for system security to be compromised. To reduce this potential, the system manager may assign a maximum time that an inactive command interpreter session can exist before it is terminated. All idle CI sessions which are inactive for more than that time period will then be aborted. Optionally, users may also configure their own idle session timeout value so long as it is shorter than that which is defined by the system manager.

The logon option in the UDC facility is a powerful tool to direct users into a predefined environment; however, if something should go wrong during the initiation process a user can be left in the command interpreter. A recent enhancement will allow the system manager to cause those user sessions to be aborted if the UDC initiation fails at the account or system level.

UDCs can also be used to intercept a users request for system functionality via an MPE command; however, UDCs can not intercept the same functionality done through the COMMAND intrinsic. Two of the new features will allow the system manager to globally disable individual MPE commands. The command may be disabled from the COMMAND intrinsic only or from both the COMMAND intrinsic and the command interpreter.

New Logging Features:

The defense of any computer system is not complete without an auditing facility. The auditing facility provides the system manager with a mechanism to monitor activity on his system. As was discussed in a previous section, MPE logs many security related events. An analysis of these events can provide a system manager with enough detail so that he can interpret system access activity.

Two new logging events have been added to the system logging facility. They are file open logging and command logging. Activation of file open logging will cause all FOPENs or

optionally only FOPEN failures to be logged along with relevant file statistics. This event when combined with FCLOSE logging can be a powerful tool in tracking file alterations.

The other new logging event records attempts to access MPE commands. The system manager may determine the set of MPE commands which he wishs to monitor. This log event will be useful in determining unauthorized access to sensitive commands.

Over the years security on MPE has become more robust as the system has evolved. As we have just seen, the recent set of new enhancements are an example of how MPE V is changing. Undoubtedly, as MPE V and MPE XL continue to evolve so will the need for better and more secure methods of insuring the security and integrity of the HP3000.

Evolving MPE Security

The next generation of MPE, MPE XL has been designed to operate on entirely new hardware. This necessitated the need for an entire new system architecture on which to operate. This new architecture provides a tremendous base upon which MPE security can be expanded.

Running on a different system architecture does not mean that the system security on MPE XL is different. MPE XL contains all of the protection mechanisms and features that make up MPE V. The different architecture merely gives MPE XL the opportunity to be more powerful than MPE V while still retaining the features which have maintained the security and integrity of MPE V.

The most important security related aspect of the new system architecture will be the multiple ring structure. MPE XLs architecture will provides multiple privilege levels upon which the software may run.

Three of the most important levels will be the non-privileged level, privileged level, and the OS level. The outer ring will contain the non-privileged level; it is where user programs will normally run. The next level in will be the privileged level. This level will initially be equivalent to the privileged level on MPE V. The inner most ring will contain the OS level. This level will be the most secure since it is where the MPE XL operating system will run.

Another important security related aspect of the system architecture is the protection ID (PID) which is associated with each process and each virtual page of data. Processes attempting to access a page of data may only do so if their PID's match. Enforcement of the protection ID is done in the hardware.

From this description we can see that the MPE XL operating system is well on its way towards being a very secure system. It builds on the years of experience that has come from the previous generations of MPE. Secondly, it contains the extensive set of protection mechanisms and features already associated with MPE V. Lastly, MPE XL runs upon a new system architecture which provides a secure base upon which system security can grow.

Conclusion

Our brief look into the MPE operating system has shown us that many of the concepts which are used to describe computer security do exist within it. The integrated nature of these

concepts merely causes them to be mostly transparent. We have seen that MPE's implementation of these concepts provide secure mechanisms to insure system security.

We have also seen that the basic MPE operating system provides security related features which protect many areas throughout the system. These features do much to control and monitor user activity on the HP3000.

Recent additions to the operating system enhance many of the mechanisms which already exist in MPE. Security on MPE will continue to evolve since the need for better and more secure methods is always present.

This evolution has already begun. Our brief look into the MPE XL architecture shows its potential for expansion is tremendous. Combining that potential with the years of experience brought to it from MPE V insures that MPE XL will be able to meet the needs of future security requirements.

Security does exist on MPE; however, a system cannot stay secure by remaining static. There are too many factors which preclude this possibility. As environments change, security must evolve to match that change. Security on MPE will continue to evolve to match it.

"The Paper Chase"

by Cailean Sherman, Unison Software Inc.

Every system manager I have ever visited has been consistently bombarded with the same two problems from each and every user. The first is: "My terminal doesn't work!" and the other is the nagging "where's my report?"

The answer to the first problem is very simple, "your terminal doesn't work because it isn't plugged in." But the answer to the second problem is the subject of my paper: how to keep your users "plugged in" with the reports they need.

Users do not care about jobs. They don't care about full data sets. And they don't even care about the absence of virtual memory. They care about their reports. Why then is report distribution such a constant headache for most shops? This paper offers some tools to insure that a company's report needs are met. The first hurdle is getting the reports created and into the spooler.

Computers are employed to manipulate enormous amounts of data and arrange it into a format which a person can easily understand. Reports are the only way to determine if the manipulation has been successful. We use sorts, merges, fcopys and custom programs to do the work, most often in batch. The goal is successful job completion run in the correct order. Huge sums of money are spent on schedulers and data base management tools.

If a data base is full, there are procedures to recover. If a job fails, there are procedures to recover. If there is a catastrophic event, there are procedures to recover. So why aren't there procedures for reports?

Recovery Procedures for Reports

I have yet to see a recovery procedure written for a report not printing. I have seen hundreds of distribution lists, but this naturally assumes that there is a piece of paper to distribute.

Operators must know how to get a new copy of a report if it does not print, or if a user claims that the data is bad. We can assume that nine times out of ten the report is correct if it prints, but *what if it doesn't print?*

Reports are vulnerable from their creation because they are spooled files which disappear once printed and may be lost with a system failure. The best environment would obviously be one where any report could be recovered from scratch. The best way to achieve this is by following these three steps.

Step 1. Determine which reports are critical.

If it is a month-end general ledger update verification, it is critical. But on the other hand, if this is a daily report of new employees, maybe if could be ignored for a day.

Operations must know which reports have to be rerun. How would you feel if you just spent four hours recovering a report which does not get picked up?

Step 2. Make sure reports can be easily rerun

If the applications are designed correctly, updating and reporting will not be done simultaneously. In most cases it should be possible to rerun the report; in those rare instances where it is not, the program must be rewritten to allow this.

Our goal is to greatly reduce the number of reports which cannot be rerun. If a report is not printed, are the users willing to give up the report? Or should a programmer spend a few hours retrieving it?

Most people shy away from duplicating data on disc and will often report on data as it is being updated. Keep in mind that disc is the least expensive resource a computer center has. Certainly, it is preferable to have production that lasts another half hour than to spend eight hours on a production rerun.

Step 3. Be sure your operators know how to rerun the reports.

There must be rerun procedures for each report. Part of implementing an application should include these procedures. The programmer has all of the information fresh in his mind as the application goes up, and if he is aware that rerun capability is a requirement, the programs will be designed to handle it.

In instances where the application is already installed and the report is a determined to be critical and that it can be recovered. this recovery procedure should be carved in stone so the operator can figure it out at 3:00 AM. As more programs are run after an error, the chances of compounding the error increase.

Paper costs a lot.

Now that the critical reports are available, should you print them?

A pallet of laser paper costs twelve-hundred dollars ($1200.00). A box of paper costs thirty dollars ($30.00). That is a lot of money for a box of paper. You can order a very nice lobster dinner for thirty dollars. (If the choice is up to you, take the lobster.)

I know computer centers where laser printers eat up a pallet of paper each week. One site in Los Angeles has three laser printers and four impact printers in one room. If they each use a pallet of paper each week, they are spending $187,000.00 per year in laser paper alone. It is safe to assume that they spend another forty thousand for the impact printers, which brings the paper budget up close to a quarter of a million dollars.

And this is only the paper cost. There are printer costs, printer maintenance costs, printer supply costs, distribution costs, and disposal costs. It is certainly a sum large enough to make a lot of lobsters nervous.

But what alternatives are there to the printed report?

Non-printed reports

Copying reports to disc and making them available online is extremely complicated. Someone has to write and maintain a system to keep track of where the disc files are, how old they are, who can look at them, how they are going to look at them and how to archive them. In most shops it is prohibitively expensive.

Using microfiche can be similarly daunting, but it can be cost effective. It is approximately half as expensive to send something to microfiche as it is to print it, and this cost goes even lower with multiple copies. For large shops this can be a great way to keep reports easily accessible for a long time, while reducing overhead by many thousands of dollars.

Spooking to tape is another option, but I have found it time consuming and frustrating to retrieve a file from tape. Most of my time is spent finding the tape to begin with, let alone retrieving the file.

The best alternative: *don't print so many reports!*

Distribution lists & procedures

There is a consistent trend seen in every data center distribution list. It never seems to get smaller. People see new reports on another's desk and suddenly it is "just what I always wanted. *Add* me to the list."

When a new application is designed, the reports are cost-justified. It takes expensive programmer time to write a report, so if it is not absolutely required it will not be written. Once that application is in place, however, the rules change. It now costs a penny a page for a new copy, doesn't it? Operator time, distribution time, printer time are not usually a valued resource.

Charging users

Chargeback systems are often the most effective way to curtail excessive report printing. A department is not going to "buy" an additional copy of a report unless it can be justified. This forces users to constantly re-evaluate their needs, and thus keep distribution lists manageable. However, some chargeback systems cannot account for exact paper usage. Usually only a line count is available.

Without a chargeback system, operations must find other methods to force users to evaluate their needs versus their wants. There are two further ways to accomplish this.

First, periodically (say every quarter), users can be interviewed and asked to justify their report requests. Operators send out the distribution list to each department and have them return it with only the reports they require.

A second alternative is more drastic, but also more successful. Don't distribute the month-end reports and wait to see who calls. But, if it's the President, there goes your lobster dinner.

Any solution you employ is going to require time and diligence — but remember, you are looking at a quarter-million dollar incentive.

Distribution requirements and practices are site-specific. However, there are a couple of "musts" which apply to all sites.

You need a list of what is supposed to print. If you know what is supposed to print, then you know when your system lets you down. This can be maintained automatically through some scheduling systems or manually with an editor.

Banner Pages

You also need some sort of system which insures that the reports come off of the printer and are not erroneously distributed. This could be solved by banner pages, although these may be hard to implement.

One way to implement a banner is to modify all of the programs that print reports to print the banner as part of the report. However, most people feel it is not worth the rewrite to gain the banner.

Another way is to modify the banner procedure in the system SL. But this method requires changing the procedure every time there is an upgrade to the operating system.

A third way is to purchase packaged software which automatically generates report banners.

You may use modified terminal type files with serial printers. This requires a separate file for each banner desired.

In conclusion

As you can see, there is no easy solution to "where is my report!" There are four steps to follow:

1. Determine critical reports.
2. Make sure critical reports can be easily re-run
3. Develop procedures for operations to follow for re-runs
4. Determine the most cost effective medium for a report.

Most shops haven't solved these problems because they are unwilling to invest the time required up front. This is how you can eat your lobster and save a quarter of million clams.

#

UNIT TESTING ON THE HP3000

BY MARK P. SHIRMAN
INNOVATIVE INFORMATION SYSTEMS INC. (IISI)
63 NAHATAN STREET
NORWOOD, MA. 02062

This is a paper on perhaps one of the most boring topics in data processing next to cleaning terminals; unit testing. However, while it may not be the most exciting nor fun topic known to mankind, it is an extremely important part of systems development. Unit testing is sometimes so tedious that it is quite often ignored or overlooked. There is however a huge benefit in performing a comprehensive unit test on every program that is developed for production. As one of the principals in a firm that specializes in HP3000 consulting, I have found that diligence with respect to unit testing can be the difference between successful implementation of a program module and a disaster. The approach that is going to be laid out in this paper can be applied to a standalone program/module or can be part of total systems development project. It is structured in its design and because of its flexibility it can be adapted to a variety of different types of projects.

I have observed that quite often in HP shops that due to either user pressure, lack of resources, or just plain laziness comprehensive unit testing is usually not performed. What is usually done is what I will term as "stub" testing. This consists of sending haphazard transactions through a program until all perceived conditions or functionality is tested. The problem with this is that it ignores the interdependent nature of the different components that make up a program. How often has a programmer lamented "Well it worked yesterday...". What usually happens is that a fix to one part of the program affected the working of the rest of the module. Without retesting from the start things start to fall through the cracks.

As part of the detailed design process, it is usually advisable to have the analyst define some initial test cycles. A test cycle represents a logical group of transactions. For instance an order entry system may have a test cycle for new orders, for modified orders, and one for invalid order conditions. The key here is define your cycles on a macro level and to try to minimize the total number of test cycles for the unit test while still covering all potential conditions. By

having some of the cycles defined up front, the programmer can get a good idea of the basic functionality of the application as well as have a foundation for their unit testing.

The test cycle will be the entity by which the unit test will be broken down. For large programs you may want to set up a chart such as the one in Exhibit I in order to track the progress of the various cycles. It is important to remember that if any condition within a cycle is not satisfied appropriately, then the entire test cycle should be restarted.

Once the programmer has completed informal testing and is ready to begin unit testing, they should sit down with the analyst, (or in the case of small HP shops, sit down with themselves) and finalize the test cycles. Following this the test conditions should be defined. A test condition represents every possible occurrence that can happen during a program's execution. Technically, there is no condition that is trivial. It is important to include invalid conditions as part of your test as well. Every test condition should fall within one of the test cycles that have been defined. A form similar to the one found in Exhibit II can be helpful in setting up the test conditions.

The next step is even more tedious than the first two, however I have found that periodic dancing, perhaps at half hour intervals, can jazz up the procedure slightly. After finalizing the test conditions, you must generate test data designed to test each and every condition. A short cut that I often employ is to try to generate data that can satisfy more than one condition at a time. Once the data is created it should then be loaded into a test data base. A backup tape of the test data base should be prepared once all the data is entered. Exhibit III is an example of a chart that can be used in tracking and preparing test data.

Using the test data as input, the programmer should then generate a complete set of expected results. Extra special care in the generation of the expected results can help insure smooth reconciliation as well as providing a programmer with detailed functional knowledge of the program/module that they may not have already obtained. Exhibit IV is an example of an expected results chart. Note how all the documentation is carefully cross-referenced so that if necessary a person unfamiliar with the all the work performed on the project could sift through the documentation fairly easily.

We are now ready for the actual testing. Each test cycle should be run separately. The actual results, which can be generated in the form of reports or Query-type listings, should be reconciled to the expected results. If there is any discrepancy, the problem should be identified and corrected and then the data base should be restored. The test cycle should be run again and again until there are no discrepancies. This

process should be repeated until all the cycles test satisfactorily.

The process that has been described here is a somewhat rigid one. However, the little extra time it takes to perform these tasks is well worth the work if it provides consistent bug-free applications to the users. It certainly beats trying to fix problem code once the programs are already in production. Depending upon the size of the program/modules that are under development, this formal approach can be modified to meet the needs of the project and the overall personality of the data processing shop.

EXHIBIT I

ABC COMPANY
ORDER ENTRY SYSTEM
TEST CYCLE CONTROL

CYCLE	CYCLE DESCRIPTION	MILESTONE DATES			RESULTS		
		Conditions Defined	Test Data	Expected Results	Test 1	Test 2	Test 3
1	New Orders	07/18/	07/20	07/22	Abort		
2	Modified Orders						
3	Cancelled Orders	07/16					
4	Invalid Conditions						

EXHIBIT II

ABC COMPANY
ORDER ENTRY
TEST CONDITIONS

CYCLE	CONDITION	INPUT REF	OUTPUT REF
1	1. NEW CUSTOMER	101	201
1	2. EXISTING CUSTOMER	102	202
1	3. DEFAULT SHIPDATE	101	201
1	4. CHANGE PRICE MONTH	103	203
1	5. ONE ORDER LINE	102	202
1	6. TWO ORDER LINES	101,103	201,203

EXHIBIT III

ABC COMPANY
ORDER ENTRY
TEST DATA

CYCLE	CONDITION	INPUT	CUSTOMER	ITEM NO	PRICE MO.	SHIPDATE
1	1	101	11600002	10101 70062	5	06/01/87
1	2	102	12005021	70003	5	05/31/87
1	3	103	11100001	10102 70002	6	05/31/87

EXHIBIT IV

ABC COMPANY
ORDER ENTRY
EXPECTED RESULTS

CYCLE	CONDITION	INPUT	OUTPUT	ORDER NO.	INVENTORY	CUST$	SHIPDATE
1	1	101	201	6001	+ 12	+3,000	06/01/87
1	2	102	202	6002	+ 15	+100	06/01/87
1	3	103	202	6003	+ 10	+250	05/31/87

INFORMATION PLANNING - A BASIC OVERVIEW

BY MARK P. SHIRMAN
INNOVATIVE INFORMATION SYSTEMS INC. (IISI)
99 RIVERMOOR STREET
WEST ROXBURY, MA. 02132

Introduction

Like many people, I am constantly asked at social gatherings what it is I do for a living. My standard reply is that I am a management systems consultant. The usual retort is typically, "Oh do you do programming?" . "Yes" I respond, "that is part of what we do, but we also perform a lot of information planning functions ". The conversation usually ends at this point with a confused look or an allusion to how good or bad the Red Sox are performing this year. The purpose of this paper is to help clarify the concept of information planning and to describe my firm's approach to the development of a comprehensive information plan. If nothing else this paper may help my conversational luck at cocktail parties and help explain to my mother what exactly it is I do for a living.

As MIS consultants, we are constantly confronted with clients that have automated systems that simply don't meet their needs as well as with organizations that are eager to computerize manual areas of their operations. Usually these companies do not know where to start and what the first steps should be. No matter how large the organization or what industry sector they do business in, our initial step usually includes some sort of information planning process and it is the cornerstone of our consulting practice. What we have found is the time and money spent up front on an information planning project will pay for itself many times later on down the road. I have been told that consultants are people that "put a price on common sense". If that is true then the following paper points out a lot of common sense ideas concerning systems development. Our experience points out however, that many of these common sense ideas are never even considered as part of the systems development project and that can lead to a time consuming and costly failure.

What is Information Planning?

Simply put, information planning is similar to the traditional strategic planning process as it relates to an organization's information. Without a lengthy definition of what information is, as it has probably been done ad nauseum, let's just assume

that it is all the key data used to run a particular business operation. The emphasis in an Information Plan will be how this data or information flows throughout an organization's workplace. An Information Plan is developed to assist a company in establishing a direction for how information is going to be processed for a period of time, typically three to five years. The Plan is a management tool which defines the information requirements for an organization, determines the application features necessary to support their business objectives, and explores how the dynamic nature of data processing technology will affect its systems. In short, the information plan is designed to offer a path consistent with the overall business objectives of a company so that the systems that implemented are consistent with the needs of the company and the resources available. Lastly, it is important to remember that the information plan should not be some "pie in the sky" high-priced report, but rather should be an action plan to insure the successful implementation of those ideas espoused in the plan itself.

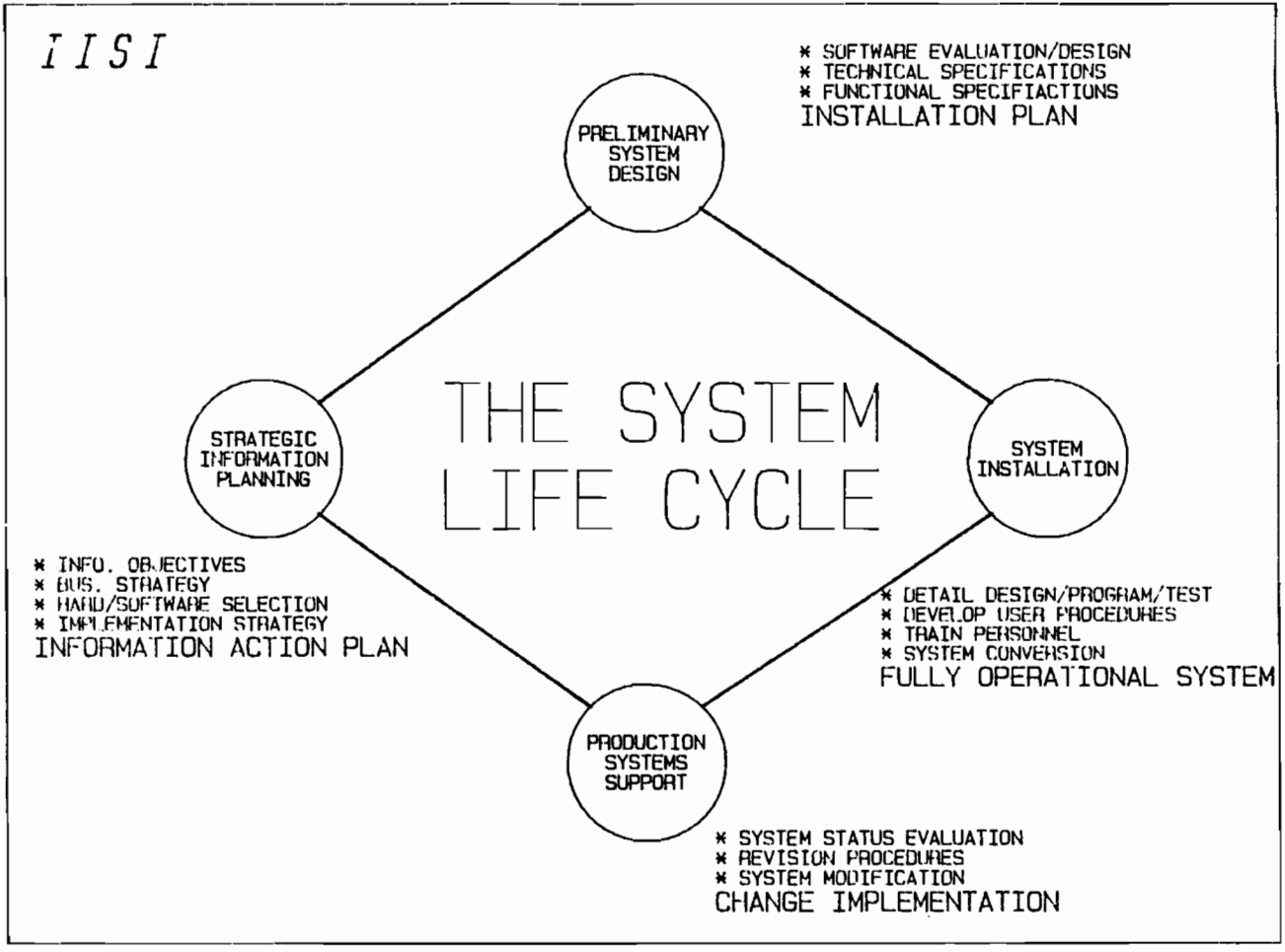
Where does the Plan fit in ?

Our firm likes to look at systems development in a modular fashion or as a "life-cycle". The life-cycle is illustrated on the following page. As you can see it is an interdependent approach to systems development with the information planning tasks critical to the following modules. The key here is that with a little extra time spent up front, the rest of the project will not only flow smoother, but the end result will meet the overall automation objectives of a firm.

What are the benefits of information planning?

A successful information planning project can yield the following benefits to an organization :

1. Support for the basic business objectives - This was touched upon briefly in the previous sections. The essence of this concept is that an information plan insures that an organization will be able to meet both the current and the future need for all types of information. Additionally, it is important that the systems implemented do not disrupt the manner in which a company does its business. Regardless of how great a system is it will be doomed to failure if it cannot adapt to to the way a firm runs its business. The integration of basic business objectives with an Information Plan will help insure that a new or modified computer system will not turn the business on it ear.


IISI
* SOFTWARE EVALUATION/DESIGN
* TECHNICAL SPECIFICATIONS
* FUNCTIONAL SPECIFIACTIONS
INSTALLATION PLAN
PRELIMINARY SYSTEM DESIGN
THE SYSTEM LIFE CYCLE
STRATEGIC INFORMATION PLANNING
SYSTEM INSTALLATION
* INFO. OBJECTIVES
* BUS. STRATEGY
* HARD/SOFTWARE SELECTION
* IMPLEMENTATION STRATEGY
INFORMATION ACTION PLAN
* DETAIL DESIGN/PROGRAM/TEST
* DEVELOP USER PROCEDURES
* TRAIN PERSONNEL
* SYSTEM CONVERSION
FULLY OPERATIONAL SYSTEM
PRODUCTION SYSTEMS SUPPORT
* SYSTEM STATUS EVALUATION
* REVISION PROCEDURES
* SYSTEM MODIFICATION
CHANGE IMPLEMENTATION

2. Improve the utilization of resources - The plan can help insure that the development effort is structured. This can help prevent redundancy of data and work effort. The responsibilities for system implementation will be clearly laid out so that the firm can plan ahead to use all available resources in the appropriate target areas. This will save both time and money.

3. Provides a firm with a flexible approach to systems development - The advantage to this is that as a company expands or contracts, the basic tenets of the information plan will accommodate these changes in such a manner that the information processing objectives of the company are still met.

4. Provides an organization with clear cost estimates - A firm will know in advance what the anticipated systems development and implementation costs will be over the years. This will be helpful in the creation of budgets and will help insure the information system provides the required rate of return necessary to meet the operation, financial , and business objectives of the organization.

In meeting the aforementioned objectives, it can be seen that spending that little extra time up front in the planning process can save an organization considerable time and money over the years as opposed to an "ad hoc" or less structured approach to information systems development.

Information planning tasks

The following outlines the tasks we usually perform as part of an information planning project. The important thing to keep in mind is that every organization is different and that the approach will be different depending upon the personality of the individuals involved. Additionally, some tasks may be consolidated with others for smaller jobs and some tasks may be omitted altogether. Remember that the overall objective of these information planning tasks is to create a plan of action not a bunch of meaningless mundane conclusions.

1. Organize the project
   Since one of the objectives of the overall project is to have a defined structured approach to information processing, it is important that the project itself be structured and planned appropriately. User committees should be set up, time estimates should be developed, and the scope of the project should be clearly defined.

2. Understand basic business strategies
   Through interviews with key management personnel and review of any business planning documents, the project team should gain a complete understanding of the basic business objectives of the company. Major characteristics of the firm should be studied with respect to competitive factors and industry trends. The key deliverable of this task should be a list of automation objectives or strategic direction memo.

3. Review current systems
   It is extremely important to gain a complete understanding of the current information processing environment. Both automated and manual procedures should be studied and reviewed. It is critical that any potential "bad habits" that exist in the current automated or manual system not be carried over into the new system(s). At this time it is often helpful to develop an information flow diagram, which can assist in the identification of potential and existing information bottlenecks.

4. Determine information needs
   In this section of the planning process detailed interviews with a company's personnel will help define those functional requirements necessary to meet their information objectives. This section will also be a key input into the development of an application strategy.

5. Develop application strategy
   At this point, potential application areas are prioritized based upon the overall business objectives of the company and upon the information needs identified in the previous segment. The project team will develop the portfolio of application system projects that will be addressed within the planning horizon, data management requirements, and potential processing architectures.

6. Hardware and software strategies
   The nature of these tasks will vary depending upon the scope of the information planning project. At the very least an approach for the evaluation of software and hardware will be outlined with the application strategy as a major input. Some projects will be more comprehensive in nature and will include a complete software and hardware selection and configuration. If this is the case there is an entire list of sub-tasks necessary for the successful selection of software and hardware.

7. Develop implementation strategy and cost analysis
   Based upon the all the data gathered in previous tasks a plan

of action is developed with an estimated timetable of events. Each of the major action plans should also contain basic cost estimates so that short and long-term budgets can developed for the various information strategies.

8. Write up Information Plan
   The plan should be written up so that all the key management of the organization can both read and understand it. It should contain enough detailed charts and information so as to completely answer any questions the management may have concerning the information strategies laid out in the Plan.

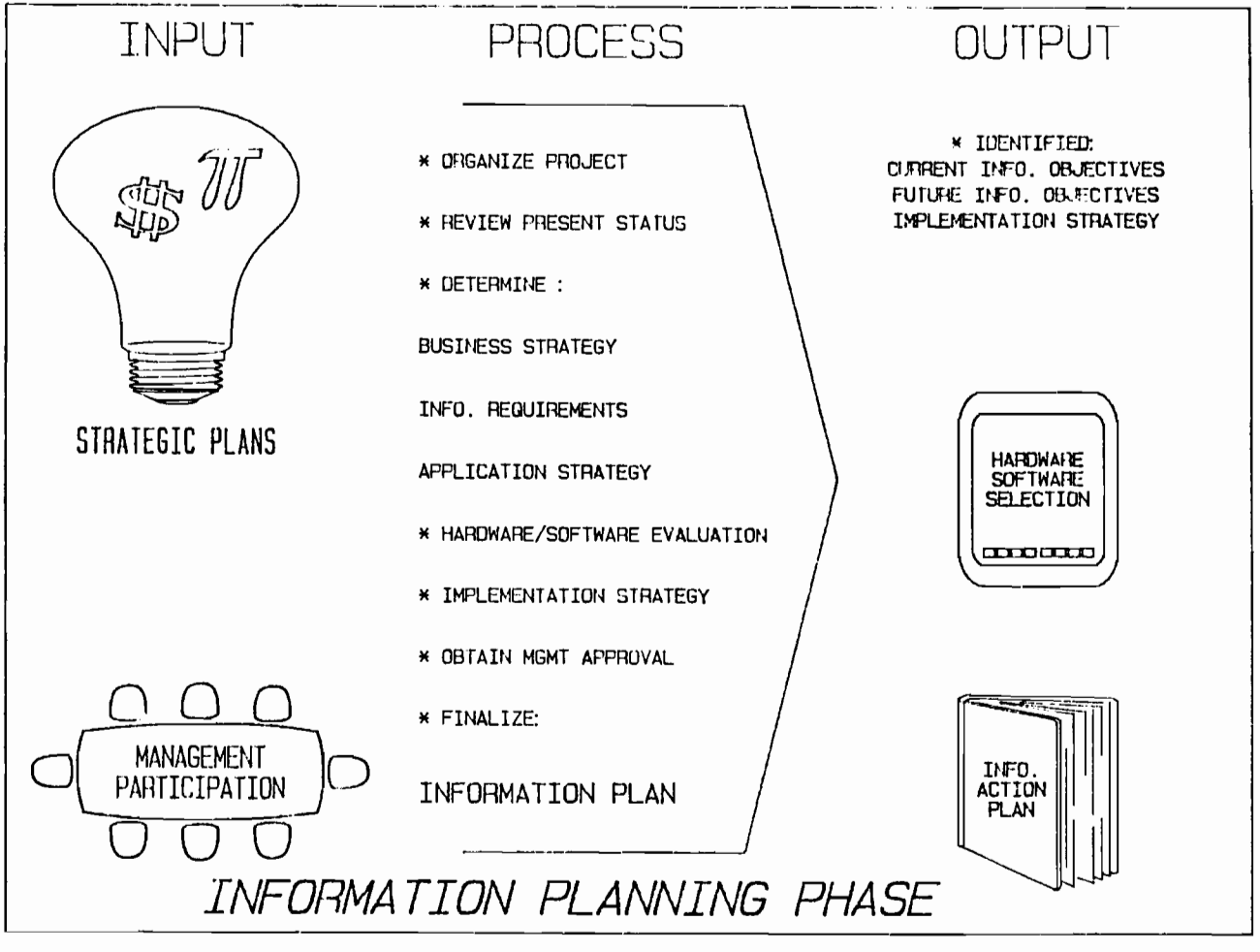
Guidelines for the information planning process

The following page illustrates the information planning process as we see it. The diagram is structured similar to a standard computer program with inputs, processing, and the output. Regardless as to how comprehensive a project is undertaken, this outline can serve as a good guidline.

Since we work with many small to mid-size companies that may have no formal strategic or business plan, it is important to emphasize the integration of the information planning process with what may be a less formal set of business objectives. No matter what size an organization may be, the owners or management usually have some concept of what they want their organization to ultimately achieve. This is why it is important to involve key management personnel in all aspects of the planning process. In this way it can be insured that the information plan can take on the personality of the company it is designed to assist and thus successful implementation will be closer to reality.

Make sure that the scope of the project is clearly defined. Nothing can put a damper on an information planning project more than a misunderstanding about the deliverable. There is a considerable amount of leeway as to what areas will be explored and to the amount of detail desired for analysis purposes. These parameters should be laid out from the very beginning.

It is important to get a complete understanding of the business that is being evaluated. The issues under consideration are not strictly technical. In fact the most important things relate to functional issues and analysis. By gaining a complete understanding for the business you can insure an information plan that not only meets their short-term information needs, but will be flexible enough to handle there operations in the future. Remember flexibility is a key ingredient in a successful information plan.


INPUT
PROCESS
OUTPUT
STRATEGIC PLANS
* ORGANIZE PROJECT
* REVIEW PRESENT STATUS
* DETERMINE :
BUSINESS STRATEGY
INFO. REQUIREMENTS
APPLICATION STRATEGY
* HARDWARE/SOFTWARE EVALUATION
* IMPLEMENTATION STRATEGY
* OBTAIN MGMT APPROVAL
* FINALIZE:
INFORMATION PLAN
* IDENTIFIED:
CURRENT INFO. OBJECTIVES
FUTURE INFO. OBJECTIVES
IMPLEMENTATION STRATEGY
HARDWARE
SOFTWARE
SELECTION
MANAGEMENT
PARTICIPATION
INFO.
ACTION
PLAN
INFORMATION PLANNING PHASE

Conclusion

No matter what size information system is going to be put in place and regardless as to what industry a business belongs to, there are benefits to be had from implementing an information planning project before proceeding with development. By doing this an organization can insure that its investment in automation has a better chance of meeting its basic business objectives with respect to both cost and functionality. By incorporating an understanding for the data processing alternatives with a comprehensive understanding for the business you can develop a a plan that is sufficiently flexible to handle the changing environment for a period of three to five years.

Optimize Productivity through Proper System Administration

by Victoria Shoemaker
Unison Software
415 Clyde Avenue
Mtn. View, California 94043

System Manager. Definition. "A person preparing for a failure or recovery from one."

System administration is a little appreciated but vital function in data processing. It is comprised of tasks which enable the system manager to handle a failure gracefully. This paper will discuss tools and techniques in the areas of:

- Backup
- Tape Library Management
- Standards
- Disaster Recovery.

Included in the discussion are forms and standards I developed which have worked for me.

Backup

Why do backups?

An old system manager once told me, "With a good set of backup tapes, I can walk away from any disaster looking like a hero." In the event of the worst disaster you can think of - the machine going up in smoke, you will be able to recover. You take your backup tapes and load them on another system. Assuming your tapes are readable, you can reload and your users will have lost only one day worth of work. This makes backup the most important part of your day.

How often should I backup?

Backups should be done on a daily basis. A full backup of your system should be done weekly. By having a complete backup done weekly, it means that you will only have to go back one week's of tapes to fully recover.

All tapes should be validated to ensure that they are readable. My SE said that it seems a waste of time to create a tape that is a backup for your entire system and not even check to see if it is good. You wouldn't deposit your emergency cash without a receipt, so don't trust the foundation of your recovery to sight-unseen tapes. On the TELESUP account there is a program called VALIDATE. It validates tapes and ensures that it can read what has been written. Validating a tape takes 15 minutes per tape, maximum. Invest the time.

It has also been my experience that one backup a month should be rotated offsite and retained for a period of one year. This gives a snapshot of your system at least once a month.

How should I do backups?

I do my backups in a job stream. I like this method because along with the SYSLIST showing all the files on the tapes is date and time the SYSDUMP was done. I have two job streams, a partial dump and a full dump stream. I do partials Monday thru Thursday and do my full dump on Friday. The job streams look as follows:

```
!JOB FULLDUMP, OPERATOR.SYS
!
!COMMENT ****************************************
!COMMENT PURPOSE:  THIS JOB BACKUPS ENTIRE SYSTEM
!COMMENT           BACKING UP THE SYS ACCOUNT FIRST
!COMMENT RECOVERY: SAVE TAPES AND CALL S.M.!
!COMMENT
!COMMENT MODIFIED:
!COMMENT VAS 1/1/87 - INCLUDED VINIT TO CONDENSE
!COMMENT
!COMMENT ****************************************
!
!COMMENT ****************************************
!COMMENT ALLOCATE TAPES
!COMMENT ****************************************
!
!RUN TAPESREQ.TAPES.CCC;INFO="FILE MONTH0;NEW;SAVE"
!
!COMMENT ****************************************
!COMMENT DO BACKUP
!COMMENT ****************************************
!
!FILE T;DEV=TAPE
!FULLBACKUP *T
!
!COMMENT ****************************************
!COMMENT VALIDATE TAPES
!COMMENT ****************************************
!
!RUN VALIDATE.UTIL.SYS
Y
Y
Y
!
!COMMENT ****************************************
!COMMENT SCRATCH TAPES (WEEKLY)
!COMMENT ****************************************
!
!RUN TAPESSCR.TAPES.CCC
N

!
!COMMENT ****************************************
!COMMENT CONDENSE
!COMMENT ****************************************
!VINIT
COND 1
EXIT
!
!EOJ
```

ZERODUMP Job Stream

```
!JOB PARTIAL, OPERATOR.SYS
!
!COMMENT ****************************************
!COMMENT PURPOSE:  THIS JOB BACKUPS FILES MODIFIED
!COMMENT           SINCE LAST FULL DUMP
!COMMENT
!COMMENT RECOVERY: SAVE TAPES AND CALL S.M.!
!COMMENT
!COMMENT MODIFIED:
!COMMENT VAS 12/1/86 - INCLUDED DAILY TAPE SCRATCH
!COMMENT
!COMMENT ****************************************
!
!COMMENT ****************************************
!COMMENT ALLOCATE TAPES
!COMMENT ****************************************
!
!RUN TAPESREQ.TAPES.CCC;INFO="FILE PARTIAL;NEW;SAVE"
!
!COMMENT ****************************************
!COMMENT DO BACKUP
!COMMENT ****************************************
!
!FILE T;DEV=TAPE
!PARTBACKUP *T
!
!COMMENT ****************************************
!COMMENT VALIDATE TAPES
!COMMENT ****************************************
!
!RUN VALIDATE.UTIL.SYS
Y
Y
Y
!
!COMMENT ****************************************
!COMMENT SCRATCH TAPES (DAILY)
!COMMENT ****************************************
!
!RUN TAPESSCR.TAPES.CCC
Y

!
!EOJ
```

Daily Backup Job Stream

The result of the above job streams is reliable backup tapes from which I can recover. These tapes are properly labelled and safely kept in the event of failure.

Cold load tapes

In addition to my backups I create 2 cold load tapes whenever I make a configuration change or install a new mit. I store one of the cold load tapes onsite and one offsite.

Tapes	Just one word of advice gathered after 10 years of working with tapes - do not use cheap tapes. It is not worth it. There is nothing more frustrating than having to do an entire system backup because of bad tapes.

It is obvious that even if our backups only took one reel we are going to have a few reels (4 - daily, 4 - weekly, 12 - monthly = 20 reels) and not many systems only take one reel to backup. It is important that we keep track of what is on our tapes, so when it is time to recover we know which tapes to mount. What we need is a tape management system.

I have developed a manual system and used a number of automated systems, but they all have the same basic principles in common. We need a way of tracking reels - all tapes look alike. We need to know about logical groups of reels; to use IBM terminology - data sets. We need a way to rotate reels back into the scratch pool when the data has expired.

Tracking reels

Simple, number your tapes. I number my sequentially. You can assign number by machine or reel size. Remember to number both the reel cover and the reel itself. Thus when the reel is on the drive, you will still know what number it is.

I put my tapes on the rack in sequential order, scratch tapes and tapes in use mixed together. I am currently using an automated system which keeps track of what is on the reels, but my manual system color-coded scratch reels with a colored dot. So even using a manual system, I was still able to differentiate between scratch tapes and tapes in use.

Tracking data sets

With an automated system, tracking data sets happens automatically with tape use. I backup my automated system with a simple manual system.

Every time a reel is used it is logged in my revised console log (sample line below) and the tape is labelled providing all important information. The minimum information on a label is as follows:

- Date Created
- Drive Created On
- Data Set Name
- Volume Number
- Total Number of Reels in Data Set
- Operator Initials
- Retention Period

Before a volume set is created the labels are created. The label is put on the door and the tape is mounted. Upon dismount, the label is put on the tape. This prevents "bad" tapes with "good" labels.

	"Good" tapes are returned to the rack after removing the write ring. If we are using my manual system of tracking scratch tapes, the colored dot is also removed.
Scratching tapes	In an automated system, tapes scratching happens automatically. To scratch tapes manually, I go through the tapes once a month and examine retention periods and "scratch" where appropriate, i.e. put on the colored dot and remove the label from the reel.
Automated tape librarian systems	There are a number of library systems available, even some free ones on the CSL tape. All do the functions described above. I found that the break-even point for me was about one hundred reels, but depending on your use of tapes it may differ for your shop.

Standards

My feeling about standards is to use them sparingly. I use them only as a convenience and in case I am hit by a bus tomorrow. I'd like people to be able to attend my funeral instead of having to stay at work trying to get things back in order.

I have set up standards and procedures for:

- Backup
- Account structures and naming conventions
- Batch processing
- Problem/resolution tracking
- Disaster recovery

I'll share my standards and procedures for your use. Use what is helpful and throw away the rest. It either fits or it doesn't.

Backup

My standards are do backup daily and a full backup once a week. All backup tapes are to be validated to ensure that the tapes are good. Rotate the first full backup of the month offsite. Retain those tapes one year. All other full backup tapes are retained one month. Daily partials are retained one month.

This ensures a year's worth of backup tapes to pull from. I heard a horror story which started me doing this. There was a shop who, unbeknownst to them, had head alignment problems with their tape drive. No tape errors occurred when they created or validated the tape. One day during a normal PM the CE realigns the heads. Three days later they have a head crash on LDEV 1. They try to reload from last week's backup, but no go. They try a full month's worth of backups, but alas no luck. In short their system manager did not walk away from this failure a hero.

My scenario would have saved them. Such is 20-20 hindsight.

Accounting structure

When I have a chance to voice an opinion, I find this simple structure works best. Account names match their use. Groups:

- PUB (ACCESS = (R,X:AC;W,L,A,S:GU)
  Contains all production programs and UDCS.
- DATA (ACCESS = (R,W,X,L,A,S:AC)
  Contains all production data files.
- JCL (ACCESS = (R,X:AC;W,L,A,S:GU)
  Contains all production JCL files.
- SOURCE (ACCESS = (R:AC;W,X,L,A,S:GU)
  Contains all source files for production programs.

- DEV (ACCESS = (R,W,L,A,S:AC)
  Contains all files under development.

Some comments on the above. I like having a development account as opposed to having a development group. This allows a test data base and a true test environment matching the production environment. Also, I like having a PROG group where my program files reside as opposed to having my programs in the PUB group. This gives me flexibility if I decided to use SL's. It also allows me to allow only execute access to these files as program files contain data base passwords which can be read via an editor if I have read access. The choices are yours.

File naming conventions

I name all files for an application with a three letter mnemonic. For example, if the application is an AP system, use the mnemonic APS. Then follow with an unique three digit number. Then a letter identifying the type of file; J - JCL, S - Source, P - Program file, D - data file, U - USL, etc. This convention allows me to look at all files associated with this application:

:LISTF APS@

Or look at all program files for this application:

:LISTF APS@P

I think you get the picture.

Batch processing

The minute you have a part-time operator to help you with your backup at night, batch processing miraculously appears. Users have this one report that they need run. Could you run it at night?

The batch processing goes through three growth phases:

- "Please stream it at night" phase
- "Please stream this every Monday after backup" phase
- "Stream this on the third workday after PAYROLL and if the moon is full and ..." phase

The first two phases can be easily handled via manual methods, the last requires an automated system. There are numerous available - Check *Interact* for advertising.

I have developed a little form which handles most requests (Figure 1). The form tells the operator the name of the JCL file to stream, when to stream it and recovery instructions. These forms are filled out by the user and submitted to the operator. Upon completion, the operator keeps one copy and returns the form to the user.

Problem/ resolution tracking

Despite our careful preparation, problems do occur. It is important that they and their resolutions are tracked. Tracking is done for a couple of reasons:

- Documentation
  The symptoms, the problem, and the resolution are documented so that others may benefit from your experience. Trends can also be analyzed.
- Responsiveness Tracking
  Tracking of how quickly and effectively a problem is resolved is helpful. Many shops guarantee a certain percentage of uptime. Your SE/CE guarantees response within a certain time. To ensure that everyone is living up to their obligations, we track.

Figure 2 shows a sample problem tracking form. The important elements of the form are:

- Date/Time
  The form should show the date and time the problem is reported and resolved.
- Problem
  The form should describe the problem. Identify the faulty equipment/application. If the problem involves hardware the serial number of the faulty hardware should be recorded. The response center will ask you for the serial numbers. If software is involved, then record the current version of the operating system and software product.
- Error Messages
  The form should record any/all error messages completely and their respective error numbers. If there are any error lights flashing, etc., record these.
- Contact
  Record who is having the problem and where and how they can be contacted.
- Resolution
  Record when and how the problem was resolved. Try to group the resolutions - hardware problem/fix, software bug, user misunderstanding, unable to duplicate, and documentation problem. If you had to cold load, warmstart or recover from tape, make sure to indicate when and what tapes where involved if any.

A copy of the completed form should be kept with the system. A copy should be sent to all parties involved with the problem and its resolution.

System managers all have their own nightmares about the worst thing that could happen. Mine goes like this. It is yearend. It is second shift, backup is being done after closing the books and we have a system failure. No problem, but while coming back up we have a head crash on LDEV 1. Not a pretty picture. The operator upset by noise drops his cigarette into the 600 LPM printer he was loading paper into. A small paper fire starts, but our quick operator puts it out with the fire extinguisher in the computer room. Your mission, recover from this mess.

My contention, of course, is that our purpose as system managers is to be prepared for this moment when it happens. In order to be prepared for this moment you are going to need:

- A hardware disaster recovery plan. A plan on how you are going to replace LDEV1, the line printer and any other equipment damaged by the smoke.
- Backup tapes. "Good" tapes from which to reload your system.
- A plan by which your operator will have contacts and phone numbers of all the people that will need to be involved in this recovery. People's names with work extensions is not good enough.
- An updated resume, in case you cannot walk away from this disaster a hero.

A disaster recovery plan is a topic unto itself. At a bare minimum you are going to need procedures in place regarding what to do in case of a real emergency, fire, earthquake, etc. Information about contacts such as SE/CE and key contacts - system manager, backup system manager, and boss with home phones. Up-to-date information about where your latest backup and cold load tapes are and their volume serial numbers.

For more extensive recovery plans, see your auditor or read about disaster recovery in back issues of *Interact*.

Conclusions

Granted that system management is a thankless job. My hope is that through the use of simple techniques and procedures, system administration will not be a painful job. Preparing for disaster will ensure that if the disaster occurs, recovery will not be a task akin to being slowly eaten by ants, but a smooth and easy task.

Preparation is easy:

- Make backup tapes and validate them
- Develop a simple tape library so you will be able to find the tapes when you need them
- Develop simple standards so you know where things are and have a standard way of handling situations
- Develop simple procedures for handling problems
- Develop a disaster recovery plan so the disaster will not catch you unaware

Good luck and may disaster never strike.

Victoria Shoemaker works for Unison Software, a data center management software vendor. As Customer Support Manager, Victoria manages its technical support program including designing and teaching its training classes. Victoria has 10 years of data processing experience. She has managed both large and small HP3000 shops.

She is a past president of the St. Louis User's Group and a past officer of the Interex affiliate council.

UNDERSTANDING THE IMPLEMENTATION PROCESS

Richard Sikon
Central Blood Bank
812 Fifth Avenue
Pittsburgh, PA 15219

I. Introduction

The purpose of this paper is to share some practical experiences related to implementing an Information System. Installing an Information System is more than the sum of its parts. It requires a considerable amount of knowledge about your business, cooperation among all levels of staff, good lines of communication and at times, "sleight of hand".

Technical aspects of implementing an Information System are downplayed in this paper. Instead, there is a discussion of some of the intangible aspects which play an important role in determining the ultimate success or failure of a project. These include the philosophical, psychological and managerial issues related to the implementation process.

The most underrated aspect of installing an Information System is the human aspect. Winning confidence early on in the game and creating a sense of ownership is essential. Communicating the role and philosophy of the Information System department produces better working relationships.

MIS is a service oriented department. Not only should we be installing user-friendly systems, but we must be friendly to our users. A good Information System cannot be developed in a vacuum and it also cannot be implemented in one.

DON'T RE-INVENT THE WHEEL

This paper is written from the perspective of implementing a packaged solution, as opposed to a product development point of view. If a package is available, then choose it. System development, even with the advent of fourth generation programming languages, is both costly and time consuming. Today building a system should be like putting together a stereo system. Rather than inventing, you have to locate all the parts and know

how to make them work together. The qualities that make a person a good system integrator and the system integration approach are a few of the topics that I would like to discuss today.

II. Background

First a little background information about myself and about Central Blood Bank. Central Blood Bank processes over 140,000 volunteer blood donations each year and distributes the blood products to 32 hospitals in the Pittsburgh area. Central Blood Bank is a member of the Community Council of Blood Centers. The CCBC and the American Red Cross are two of the major players in the blood banking business. Most hospitals no longer find it cost effective to draw and process blood independently anymore.

I've been actively involved in a computer related career for nearly 15 years. During this time I have been involved in numerous aspects of design, development and implementation of systems. I joined Central Blood Bank four years ago as Director of Information Services. By nature I am a generalist. I believe that productivity can best be achieved by understanding the "big picture".

When I joined Central Blood Bank a decision had already been made to replace the existing blood banking package. This was necessary since the vendor from which the package was originally purchased was no longer a viable business and it had become increasingly difficult and expensive to support the existing system. In retrospect, I see this as one of the first important factors in successfully implementing a new system. That is,

A SENSE OF URGENCY MUST EXIST

This urgency must be understood throughout the organization. This helps establish deadlines that cannot be easily changed. It also helps establish momentum and creates a need to maintain it.

MOMENTUM MUST BE MAINTAINED

Once a project is started you must continuously work to minimize all other distractions. Overcoming the original inertia is extremely difficult and once momentum begins to build, it must be maintained. It is counterproductive to juggle projects.

A blood banking system consists of several major functional components. Information must be maintained on blood donors and blood donations. Blood donors must be recruited and blood donations collected. Blood donations are manufactured into component products, laboratory tests are performed and products are labeled and released to be distributed to hospitals. It is necessary to be always able to track units from the time of donation through time of transfusion to a patient. If there are complications resulting from a transfusion, the original donors must always be able to be located.

To find a suitable replacement for our system, the following table was developed to help analyze all possible commercially available systems. (See Figure 1).

Obviously, establishing good selection criteria is essential in locating the right system. The table devised in Figure 1 is simply a way to uniformly evaluate various alternatives.

EASE OF IMPLEMENTATION IS DIRECTLY RELATED TO MAKING THE RIGHT CHOICE

To successfully implement a system you must be willing to do your homework. If you feel confident about the decisions that have been made, it will be easy to convey this enthusiasm to others. Understanding the system is time well spent. Numerous pitfalls can be avoided during the implementation process and in later operation, by simply understanding the system well.

III. The Implementation Plan

"Cheshire puss," she (Alice) began ... "Would you please tell me which way I ought to go from here?" "That depends on where you want to go to," said the cat.

....Lewis Carroll

As the project manager, you must know where it is you're headed and how you'll get there. A detailed plan must be produced, specific milestones must be established and this information must be communicated to all the people involved. The charts and the schedules are an obvious necessity in the implementation process, and I will not dwell upon them. Instead, I would like to focus on the human interactions that ultimately determine the success or failure of the project.

Figure 1

BLOOD BANK SYSTEM UPGRADE

SUMMARY OF ALTERNATIVES

Rating Scale

1 Bad 2 3 4 5 Good

Major Selection Criteria	Organizations Vendor 1	Vendor 2	Vendor 3
Organization -Stability -Support capabilities -Methodology	2	3	3
Product -Flexibility -Maintainability -Comprehensiveness	3	2	3
CPU -Reliability -Support -Expandability	4	4	2
Operating System & Language -Efficiency -User friendly -Maintainability	3	4	1
Data Management -Reliability -Standardization -Support	3	5	1
Risk Factor -Weighting factor -Collective effect of criteria	1	3	1
TOTAL POINTS SCORED:	16	21	11

COMMUNICATION

An effective mechanism to communicate information and resolve problems is another essential implementation ingredient. Establish an implementation task force which will share the responsibility for the project. The task force must consist of decision makers that represent the various areas affected by the system and they must meet on a regular basis until the project is completed. Positive features of the system should be continually promoted within this group. Limitations must be de-emphasized. The project task force must understand the merits of the new system and they must be able to convey this to their staff.

THE SYSTEM MUST BE SOLD AND YOU ARE THE SALESMAN

Each individual on the project task force must understand that success ultimately will rely on the acceptance of the system by the people that actually use it. A good system can be implemented poorly and it will fail. A poor system can be implemented well and it will be a success. The people that will use the system must feel responsible for its success. A sense of responsibility and ownership is an important part of the process.

A SENSE OF OWNERSHIP MUST EVOLVE

Growth is a process - implementation is also a process. A sense of ownership for the system will not occur immediately. It must be slowly nurtured. It will require patience and time. There is a natural order to these things just as it exists with growth. Your ability to help the process along reflects on your ability as a Project Manager.

Implementation is the culmination of all the planning that has preceded it. It is the act of accomplishment which results in a concrete measure of success or failure. A successful implementation is ultimately linked to user acceptance of the system.

Implementation is an on going process. Standards and guidelines are essential for growth to occur in the Information Services department. It is important to have an organized approach to control requests to modify the system. At Central Blood Bank a system investigation request must be submitted for each request to modify or enhance the system. These requests are evaluated and prioritized. (See Figure 2)

Figure 2

BLOOD OPERATING SYSTEM
System Investigation Request

Requesting Individual: _____

SIR # _____ Date Requested __/__/__

Description _____

_____

_____

Screen XREF: _____/_____/_____/

Report XREF: _____/_____/_____/

Why Requested: _____

_____

_____

Tangible Benefits: _____

_____

_____

Intangible Benefits: _____

_____

_____

Supervisor Acceptance _____Date __/__/__

System Mgr. Acceptance _____Date __/__/__

The System Investigation Request is the first step in making it generally understood that a cost benefit justification is required for each request. Requests are selected based upon their overall benefit to the organization.

IMPLEMENTING AND DEVELOPING DO NOT MIX

You should try to avoid considering requests to change the system while you are still in the implementation phase. Development is another layer of complexity that can only further complicate the implementation process. A good rule to establish is to not change the system for a period of six months to one year from the time the system is placed into production. In Central Blood Bank's case this was a contractual requirement within our software licensing agreement with the software supplier. On numerous occasions this enabled us to hold the line on modifications. The end result which occurs in most cases is that people learn to work within the limits of the system. It is amazing how many requests to modify or enhance are no longer needed once the system is in actual operation for a period of time. People can be quite adaptable if they are given the opportunity.

Proper training will also play a key part in a successful system implementation. Our approach was to train the trainers. These individuals were then responsible for training the staff within their department. This was beneficial because it helped distribute a rather substantial training effort among many different departments while at the same time creating a sense of responsibility at the department level.

There are numerous other implementation related issued. One that I feel deserves comment is the concept of parallel operation. Parallel operation is a method to verify the accuracy of the new system. It involves maintaining the original and the new system in a parallel production mode for a specified period of time.

I have never been fond of this concept. Placing a large system into production is one of the most stressful times for any organization to endure. Telling your users that they must perform duplicate system related work is not a good way to win friends and influence people.

Instead of parallel operation, I would much rather see the effort spent on system verification. With this approach, you must perform extensive functional and integrated testing prior to placing the system into production. A training system and a training data base can be established so that each department can verify their portion of the system. The end result of the system verification process should be a high degree of confidence in the new system. The advantage of this approach is that it can be scheduled and controlled. If you have done a good job at system verification then parallel operation may not be necessary at all. However, if you feel that a period of parallel operation is an absolute must for your particular application, then my best advice is to make that period of time as short as possible.

IV. The Essence of the Implementation Process

In the implementation process you must ...

- -understand the business
- -understand the system
- -understand where you're headed
- -understand the people you must work with
- -understand your limitations

In the implementation process your role is that of a guide. There are many paths to take. You must help people choose the right ones.

AVOID UNNECESSARY COMPLEXITY

The best solutions are always the simplest solutions. If Henry David Thoreau was a MIS manager today, as unlikely as that may be, he would still be saying, "simplify", "simplify". Avoiding unnecessary complexity is the objective of normalization in the mathematical sense. It applies equally well to data bases, to systems and to life.

Implementation is about communication and about adaptability to circumstances. It is important to make people aware of the philosophy of the Information Services department. The following tables describes the Information Services department role and philosophy at Central Blood Bank.

The Role of Information Services

* Provide service
* Manage technological change
* Act as a centralizing force
* Provide information control

Operating Philosophy

* Distributed processing
* Self-reliant user departments
* Cost/benefit analysis
* Work smarter

Implementation is about problem solving. Generating awareness is the first step in solving a problem. To solve a problem a context must develop and a common level of understanding must evolve. Developing a common frame of reference is best achieved by developing a written plan.

V. Working Smart

"It is easier to ask dumb questions than it is to fix dumb mistakes. . . ."

....F. Alfredo Rego

Implementing systems is also about working smart. Your staff is your most valuable resource. It is wise to invest your time coaching them and providing them with the tools that they need to perform their jobs. In the long run, this can be one of the most valuable and rewarding investments of your time. It is an investment that can pay substantial dividends.

I'd like to leave you today with some timeless truths about performing at your best. These are disciplines that Peter F. Drucker, America's most esteemed management thinker, has identified that all effective executives share.

1. Effective executives know where their time goes. They continually diagnose how they've been spending their time and are quick to prune unproductive activities. Yet they aren't afraid to devote lots of time - even if it seems excessive - to issues that matter.

2. Effective executives judge themselves by results - not by time or effort expended on a certain task. They always start out by asking themselves "what's expected of me?" rather than worrying about the techniques or tools they'll need to get there.

3. They build on strengths - their own and those of the people they work with. They focus on the opportunity in their staffing, not on any problems or deficiencies. Yet they are quick to fire anyone who consistently fails to perform.

4. They concentrate on the few major areas with the highest potential for achieving outstanding results. They force themselves to set priorities and abide by them. They know they have no choice but to do first things first - and ruthlessly ignore everything else competing for their attention.

5. They make effective decisions which means a few fundamental decisions rather than a lot of fast, razzle dazzle ones. They know that to make many quick decisions means to make the wrong decisions.

To Peter F. Drucker, old fashioned effectiveness is what is really important. Effectiveness depends on concentrating on the right things to get the right results. The most effective executives know that single minded concentration on one task at a time is as close to a "secret" of effectiveness as you will find, and the issue of performance invariably boils down to being conscientious.

DETAILS, DETAILS, DETAILS

MIS is a very detailed oriented occupation. Ultimate success is rarely achieved through quick fixes and flashy solutions. Instead it is a basic concern for details and a step by step approach which produces the most reliable results.

The nature of MIS is such that there will always be more projects waiting to be done than can be accommodated with the resources and time available. You must choose your priorities wisely. Do the things that will provide the most benefit to the organization and balance this with the things that you are most qualified to do. Understand your limits and work within them. In the final analysis almost anything can be done. But the real question is - is it worth doing - and what is its relative importance in the scheme of things.

STRATEGIC PLANNING IN SMALL MIS SHOPS

TERRY W. SIMPKINS
Spectra-Physics, Inc.
Retail Systems
959 Terry Street
Eugene, Oregon 97402

Strategic Planning, what is it, why is so much attention being paid to it these days, why should it be done, and assuming that it should be done, how does one go about it? All very good questions that do not have obvious answers but need to be understood by every MIS manager; especially in small shops where resources are extremely limited.

WHAT IS STRATEGIC PLANNING ?

This paper does not pretend to be the definitive explanation of the topic, but rather to highlight the experiences of one small shop that has gone through the exercise. The state of the literature on strategic planning for MIS outlines a massive project covering every conceivable aspect of systems, lasting well over a year, employing several dedicated people, and having a considerable price tag. The results of this process is a document of considerable weight and volume. Every detail of future systems would be explained, and a considerable amount of the initial design work for these future systems might well be included. Detailed forecasts of the expected volume of transactions going through the systems may be outlined along with pages of explanations for the volume changes. While this may be possible (even reasonable) for a large installation (say Ford Motors), the concept is laughable in a small shop of say five people. The small HP3000 shop normally consists of one to six people and is completely buried already, the thought of adding something of this magnitude is simply not within reason. Faced with this situation, we undertook the task of developing a workable strategic plan. However before we could begin the process, we had to develop the methodology that would be used. This is a description of that methodology and some of the reasoning that went into it.

For the small shop it is very important to plan, perhaps even more important than for large installations, since we as small entities are more sensitive to change. But we must also consider the very limited resources available to us and keep the amount invested in the planning process in perspective. Because of this our effort will by definition be much smaller and of less detail. The crux of the issue is to develop realistic expectations of the planning process

and to resist the desire to develop an all encompassing document that is all things to all people. Define exactly what you expect the process to produce and focus on the activities that will address those goals, and stick to them. In my opinion, realistic expectations are developing an outline of what the upper management of your company (or division) expects the environment to be for the next several years, and what are the types of systems required to support that environment. Any more detail, and the level of confidence drops quickly.

WHY PLAN?

If you are considering the development of a strategic plan, I assume that you are aware of the reasons planning is important. Perhaps you are here because your boss has told you that you are going to develop a plan. Which ever is true, a quick review of the major reasons that I see for planning is in order.

First, businesses change over time, as do the systems requirements of the business, while installed software stays perfectly constant without human intervention.

Second, non-trivial computer systems take a considerable length of time to develop and install. This may include definition, design, selection, coding, training, etc. Whether you create or purchase the software, it is not an overnight project.

Third, all resources are finite, and we want to make the best possible use of them. If we can avoid spending valuable resources on short term needs in favor of addressing needs that will be with us for a longer period of time, we have probably gotten more value for those resources. Likewise if we address the most important needs of our users first, we develop credibility with management and that has several benefits.

Fourth, by having a better understanding where we are headed we minimize "mid-course corrections". This saves valuable time and effort and gives the appearance that we know what we are doing.

Fifth, we generate enthusiasm and user buy-in when they understand the direction systems are taking. Even if they do not totally agree with the direction, they will be more cooperative if they at least know what is going to happen.

There are more reasons why a strategic plan should be developed, but as MIS professionals, you should already

understand them and if you do not there is plenty of published material on the subject.

PURPOSE OF THE MIS PLAN

First, the MIS plan should provide an overview of the state of the current systems and how they got where they are. This historical perspective may not add value to the future plans, but will provide good background understanding for why we are planning now and some of the problems that can be avoided by planning.

Second, the plan should provide insights into the direction of MIS both from a broad general perspective as well as some of the specific needs that are identified. And a general time frame for when these changes should be in place.

Third, you need to measure how far you currently are from where you want or expect to be. This provides a basis for priority setting and resource allocation. It also prepares management for the requests you are going to make and gives them a measure of how reasonable and realistic the plan.

Fourth, once the target is established and you understand how far away you are from that target, you can outline what is needed to move toward your goals and how quickly you can expect to get there.

COMPOSITION OF THE MIS PLAN

I. Introduction/Narrative Summary

In order to provide a basis for the plan a brief historical recap of the current set of systems is included to help define where we are today. Trends are discussed as well as the reasons for the major decisions that have been made. The purpose is not to justify or condemn, but to provide better understanding of why we are where we are. The next section covers where we want to go in very general terms. This is the visionary portion of the plan, my chance to gaze into the crystal ball and present my vision of what MIS should be and the role it should play in the business. Included in this are the type of resources required to support that vision; people, hardware, software, business commitment, etc. All of the requirements may not exist, but I can describe what I need anyway. A review of the current role of MIS is included to compare and contrast the current environment with my vision of the future. Next, a review of the MIS organization. Describing how we are organized, the equipment currently in place and a recap of the departments strengths and weaknesses. Finally, a brief comparison with other MIS departments in organizations similar to ours. Size

of staff, machine capacity, services performed, and budget as a percent of sales comparisons provide a relative measure of our performance.

II. Existing Systems Profile

For each system, a narrative is created covering in detail the basic function of the system, state of the code, documentation, user understanding, MIS understanding. Also discussed are the things the system does well and the areas that the system does not address or that it addresses poorly, including major known bugs or omissions. A general description of the backlog of change requests for the system provides a basis for management to judge the validity of our assessment of the system. We then recap the expected impact of these changes in terms of cost, time, stability and probability of success.

Ideally your business has or will develop a strategic plan for the entire business; if so, you will be able to use that plan as input to the MIS plan and build from it. If not, then portions of that general business plan must be developed either explicitly or implicitly in order to proceed further.

III. Comparison of Existing and Future Operating Environments

For each functional area of the business a projection of the future environment is required if we are to understand and project the information and systems requirements. This projection must be that of the functional manager, it is not required that he/she actually create it, but he/she must agree with it and be willing to sign his/her name to it. The purpose of this projection is to compare the current environment with that of the future and to understand the impact of the changes on the information needs of the business and the ability of the current systems to meet those needs. Each functional area is divided into the processes that compose it. For example Marketing might be broken into the following subcategories: Promotion & Advertising, Sales, Market Research, and Market Planning. For each of these areas, a profile is created listing the important attributes, a description of the current environment and the expected future environment (see Form #1). The information on this form is strictly business oriented, it should be completed by the users and not consider systems at all. If a strategic plan already exists, or is being developed in conjunction with the MIS plan, this should be a part of that plan. If not, you must develop it at this point in the process, since the rest of the plan builds upon it.

The next step in the process identifies the important "information systems" used in the operation of the business. These systems are not necessarily computer systems, but rather the way information is collected and arranged or used. Systems could be defined as processes or functions (ie. Master Scheduling, Purchasing, etc.), or along the lines of the current computer software packages if that is felt appropriate. The important thing here is not the way "information systems" are defined, but the listing and explanation of the critical factors that impact these systems and the informational nature of these critical factors. The intent here is to highlight the type of information that is needed to support the future business environment. Form #2 is one method of ferreting out this information. Each critical factor is then described according to how it is impacted by the seven information characteristics listed across the top of the form. Any factor listed must, by definition, relate to at least one of these characteristics. It may relate to more than one, but seldom to all of them. These descriptions are then the basis for evaluating current systems and any alternative systems or designs (see Forms #3 & #4). Because these factors are "critical" and are impacted by certain information characteristics, they are the obvious basis for analysis and comparison of alternative systems. Of course there will be other criteria in the selection process such as cost, complexity, support available, interconnection with other systems, etc.; but this list of critical factors and the informational requirements will be a vital part of the requirements definition.

In our case a proposed replacement system had already been identified and we focused our attention on that alternative. The process being that if that alternative did not sufficiently meet the users needs, then we would start from scratch to evaluate other systems. It should be noted here that the outcome of completing Form #3 could be that the current systems satisfactorily meet the needs of the enterprise. In fact this conclusion would be expected where systems had been recently replaced. This would not mean that the exercise was wasted effort, but rather that those involved in the planning process had verified that business needs were being met. Businesses periodically examine the market segments they participate in to insure that they are in the correct ones, and should likewise examine their systems to insure that they are appropriate and providing the required information.

It is critical that users complete these forms. The MIS staff should assist to insure that a proper level of detail is included and that the descriptions are measurable and quantifiable; however, in order for there to be a meaningful

result the users must "own" the information that comes from the exercise. In my experience, the generation of the forms used in the project is very important. The format of the information gathering must assist in the extraction of the information required to construct the plan or you will find it very difficult to communicate your goals to all involved parties. The forms should lead the users through the process and force them to provide the required focus on business issues and information requirements. Another tool to help and guide the users is to create a "straw man" list of business attributes, information systems and critical factors for each of the areas. This list should not be cast in stone, but should provide "food of thought" for the users, seeds to start their thought process.

As the future information needs of the various areas are assembled, the cost of meeting the needs and the impact of not meeting these needs must be understood, along with the relative importance of the needs in order for management to make correct resource allocation decisions.

IV. Plans to Meet Future Needs

Once the future needs of the business have been identified, quantified, and ranked, a plan to meet those needs must be developed. This may include the selection process for new software, a list of modifications to current software, and an installation/project plan. Project management skills and tools play a very important role in this section of the plan. It is not critical that the actual details of the process be included in the plan, rather that the process that will be used is defined and understood so that management can buy off on it. This section will also include the sequence of projects to be undertaken. This will reduce the "mid-course corrections" caused by misunderstood priorities and increase the confidence level in the plan since management has reviewed and "blessed" it. This is the place to address the hardware required to make the software plan possible. Where possible, link hardware needs with specific projects, to reflect the true cost of various alternatives and projects. Keep in mind that hardware requirements are usually a step function and seldom follow a smooth curve. Form #5 is one method of presenting the action items required to "fill the Gaps" between the current state of systems and the state required to meet the future needs of the organization.

HUMAN RESOURCE PLANNING

As a part of your plan be sure to address the people portion of your department. This is your greatest asset and deserves attention just like your software and hardware.

Succession planning, career development and professional training are all part of a complete Human Resources Planning effort (see Form #6). Personnel planning is important for several reasons. Much time and money has been invested in your programming and operations staff to get them to their current level of understanding of your systems; by failing to provide an effective career plan for these employees you risk loosing them and having to reinvest in training their replacement. This retraining effort also has opportunity costs associated with it in addition to the cost of the training; that being the value of the projects that cannot be undertaken while this training is taking place. These costs can very often be avoided with good career planning. Remember too, that happy employees are more willing to work the long odd hours often required in our profession and are always more productive than unhappy ones. By developing your staff and expanding the knowledge base of each employee you also reduce your exposure in the event that an employee does leave. By having several people who can perform each job or support each system, you have designed a back up for everyone and you improve the quality of systems because interactions and interfaces are better understood by everyone.

As stated at the beginning, this paper is not designed to be an exhaustive study in the art of strategic planning, but rather to reflect some of the insights gained by performing it in a small shop environment. All portions will not be applicable to every installation, but the general requirements and approaches I feel are common through all companies. Feel free to use the forms and modify them to best serve your needs.

BUSINESS ENVIRONMENT

Today vs. Future

Form #1

Functional Area

Department

Business Attribute	Today	Future (5 Years)

GAP ANALYSIS --- _____ FUNCTION

FUNCTIONAL STRATEGIES	STATE REQUIRED	CURRENT GAPS	ACTION REQUIRED TO FILL GAPS

Functional Area

Department

Change in Information Needs
Due To
Changes in Business Environment

Page

Information System (Not Computer)	Critical Factor That Drives or Impacts the Information System	Timeliness How Quickly Updated & Available	Detail Level	Information Commonality Same Data Across User Groups & Which Groups	Arrangement Indexes, Sorted Order	Time Horizon How Much History is Required	Accessibility Who Can Access Data	Accuracy How Critical is Absolute Accuracy

Information Needs vs Proposed System

Functional Area

Department

Information/Functional Requirement	Ability of ASK System to Provide

GAP ANALYSIS

Information Needs vs. Current System

Functional Area

Department

Information/Functional Requirement	Ability of Current System to Provide

FORM #6

Are assumptions upon which the strategic plans are based realistic regarding human resource requirements?

Over time, what skills will become obsolete, change in nature, or be eliminated?

For what functional skills/positions are we likely to encounter a shortage of qualified candidates in the marketplace, now or in the future?

Do the present managers within the function have adequate technical/managerial skills to meet the strategic changes occurring at RSD?

What are the principal H/R obstacles to achieving the function's strategic objectives?

Are age patterns in the organization imbalanced, suggesting high future attrition or career path blockage?

Is there adequate or excessive turnover in any group, at any particular level?

Is there a proper balance (staff mix) of managerial, professional, technical, and support staff within the function?

What are the most significant skill deficiencies within the function organization? How will such gaps be addressed?

FORM #6 (continued)

Are the organization and structure and staffing of the function appropriate for the achievement of strategic objectives?

To what extent will qualifications for existing positions change in light of strategic plans? How will such changes be addressed?

Do the strategic plans/objectives call for projects or processes that have no precedent at RSD? What are the implications for staffing requirements? Design of the present function organization?

Which positions, if not filled, will have the most detrimental effect on achieving the function's objectives?

What impact would a product line de-emphasis or discontinuance have on the responsibilities of the function staff?

EFFECTIVE HP3000 ACCESS SECURITY WITHOUT MPE PASSWORDS

Kelly Spencer
State Farm Insurance Companies
One State Farm Plaza; A4
Bloomington, IL 61710

If you're not somewhat skeptical after reading the title, I'm surprised. On the surface, it doesn't seem sensible to abandon an HP3000's inherent password system in order to improve security. However, continuing to work around the inherent problems of MPE passwords doesn't seem very logical either.

A summary of my working environment should exemplify my MPE password concerns. I am the data security officer for the twenty-five HP3000s at my company's corporate headquarters. There are several hundred users in my environment, mainly utilizing the systems for application development and office automation. Because of our data center's size and complexity, we have several areas assigned PM and/or SM capability for their respective functions. These analysts protect their user IDs with MPE passwords, but a given individual's password may be the same on every system. In addition, these passwords rarely change because of complacency and/or because of the consequent job file changes required. Other system users, assigned more or less default capabilities, may or may not have MPE password protection. If they have passwords, the same difficulties are encountered when encouraging their regular change.

I feel that my company requires two general improvements to its HP system access security. First, there should be a more effective password alternative for all interactive session initiation. Second, a non-password method of authentication for all non-interactive system access should be developed.

I am aware that alternative HP3000 access security software already exists, offering encrypted password options for improved interactive session security. Features available are automatic password aging, decentralized password maintenance, minimum password length, and the requirement that the password value be different when changed. In addition, vendor passwords can also be assigned to a logon profile, rather than just one qualifier. For instance, I may have my own password for KELLY,MANAGER.SYS,PUB, while my coworker's password may be assigned to JANET,MANAGER.SYS,@.

These non-MPE passwords are unusable in the batch environment, however. Consequently, a user is required to retain at least one MPE password to offer his or her ID some level of batch access protection. Retention of an MPE password consequently necessitates retention of all of the MPE password problems.

I contend that this retention would serve to weaken my company's HP3000 access security program. Since a user would be required to remember at least two passwords for each one of their user IDs, he or she will likely change the passwords less often and write them down more often. They will also tend to assign both passwords the same value, negating the effectiveness of the non-MPE password's encryption.

A popular attempt to improve batch access protection is the dynamic insertion of MPE passwords prior to job execution. For example, after issuing the STREAM command, I would respond to a prompt(s) for an MPE password(s) to validate my usage of the ID(s) in the job file. This approach, however, still requires a second, encrypted password for more effective interactive access protection. Without this additional password, I would potentially be sharing both the batch and interactive usage of my ID with another user (perhaps users) when I communicate my MPE password(s) to him or her for the express purpose of job execution. The most important drawback of this approach, however, is that it injects an awkward and unnecessary interactive intervention into the batch environment.

The April 1987 issue of the HP Chronicle reported that the aforementioned password insertion approach has been adopted by Hewlett-Packard for inclusion in a future operating system release. I obviously do not agree with this direction, and I consider it simply an additional yet important reason for sharing an alternative batch control philosophy.

I propose that an HP3000's batch access security be managed via a data base rather than by MPE passwords. Let's say, for example, that my assigned user profile is KELLY,MANAGER.SYS, and that I regularly require the ability to stream a job using a TELESUP user ID. A security data base entry could be made to allow my KELLY,MANAGER.SYS interactive logon to stream jobs using KELLY,MGR.TELESUP.

In the above example, I would still require an encrypted, aged, non-MPE password to interactively log on as KELLY,MANAGER.SYS. However, I would be unshackled from password validation when using KELLY,MGR.TELESUP in batch, and I would be precluded from knowing how to interactively log on to the TELESUP account.

An important feature of this data base security approach would be the automatic data base entry and data base deletion of each interactive logon profile respectively added and removed from the HP system. This would allow a user to automatically initiate batch sessions with his or her interactive logon profile and would require no special data base maintenance by the security administrator. Another requisite data base administration option would be wild card usage for logon qualifiers. For example, a system administrator may then be given batch authority for @,@.@,@, rather than for all of the more specific profiles in existence.

A batch access control system such as this should improve user community satisfaction (or, at least its tolerance) with your organization's security software. For interactive access, personnel would be required to have a password, but it would be one that they exclusively select and maintain. The password would be encrypted, and the security package would possess all of the features that comprise good password management (minimum length, automatic aging, etc.). For batch access, there would be no passwords necessary. Many users would not need to request any batch authority, as they would automatically possess all such authority that they require via their own ID. Those users needing broader batch access authority than this would secure the permission of the other user ID owner(s) and have the security officer add their intentions to the security data base.

There are other features that a data base-driven batch access security system may possess, such as perhaps automatically expiring data base entries, but these enhancements are only supplements to the general philosophy I am proposing. There may also be drawbacks to this approach (such as establishing batch authority on a given machine when a batch process from another machine DSLINES to the former and logs on with a REMOTE HELLO). Borrowing from a contemporary cliche, however, if we can fly a man to the moon, there should be a technical solution for such shortcomings.

There may also exist the contention that it is financially imprudent to develop an HP3000 access security system that does not rely on HP3000 passwords. I would counter that it has been more wasteful and far less effective developing such software that does rely on MPE passwords. If users make it clear that they require better batch access security, I am confident that demand will financially encourage supply.

Hewlett-Packard has bent over backwards to create parallel MPE environments for interactive and batch modes. This approach detracts from an HP3000's access security, however. My proposal offers an alternative environment to consider. Hopefully, it will at least fuel discussion that will direct the HP user community toward a comprehensive access security philosophy that is more effective than what, until now, it has had to settle for.

MPE V to MPE XL Migration Overview

by
R. Gregory Stephens

Hewlett-Packard Company.
19111 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, 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, Database Compatibility, Source Code 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 Tools in these proceedings.

Migration Process Overview

The Migration Process consists of six stages that delineate the steps 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 Services that Hewlett-Packard is providing, as well as the areas of compatibility that are provided

by the 900 Series HP 3000 systems. This paper and past papers and presentations on various migration topics are Hewlett-Packards first steps in providing information on migration.

The second Migration Stage is ***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.

System Installation. is the fourth stage. 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. 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 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. A **Migration Planning Guide** is available to provide more detailed information about the migration process.

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 new features. This manual set will also include a Migration Process Guide 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** 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.

Planning

The goal of the ***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, called **FastLane 3000**, that helps plan the migration of applications is available. It is 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 is available in two parts, System Planning followed by Application Planning.

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 MPE V/R based systems in moving directly to a 900 Series system.

A Migration Toolset is 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 ***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. When multiple HP 3000's are involved in the migration, planning for which system each group of users will be using after the new machine(s) are installed should also be done. 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 you 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 preparation. During this stage everything that can be done on an existing MPE V/E based HP 3000 in preparation for delivery of 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. While use of the latest version of a particular language is required to recompile a program in Native Mode, the performance characteristics of an application should be analyzed before any major conversion efforts are undertaken. This is because the amount of added performance that will be achieved when a program is recompiled into Native Mode is completely dependent upon the characteristics of an application. However, all new development should be done in the latest version of a given language to provide the smoothest migration to Native Mode. When possible, applications that would normally be written in SPL should be written in Pascal/V or C.

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 utility. 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 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 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. (It is important to note that an Object Code Translated program still executes in Compatibility Mode. This is because it continues to execute against a 16 bit stack using MPE V HP 3000 addressing.) 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 a key element in our migration strategy, 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.

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. Cargnoni and I. Janet Garcia, INTEREX 1986 Detroit Conference

MPE V to MPE XL Migration Tools by R. Gregory Stephens, INTEREX 1987 Vienna Conference and INTEREX 1987 Las Vegas 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 Business Unit 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.

MPE V to MPE XL Migration Tools

by
R. Gregory Stephens

Hewlett-Packard Company.
19111 Pruneridge Ave.
Cupertino, CA 95014

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 systems. 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 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 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, Database Compatibility, Source Code 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.

If you have read the MPE V to MPE XL Migration Overview paper then the following description of the migration stages may be skipped.

Migration Process Overview

The Migration Process consists of six stages that delineate the steps 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 Services that Hewlett-Packard is providing, as well as the areas of compatibility that are 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-Packards first steps in providing information on migration.

The second Migration Stage is ***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.

System Installation. is the fourth stage. 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. 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.

Planning

The goal of the ***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'. Compatibility Mode allows object code object code developed on an MPE V based HP 3000 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 MPE V HP 3000 instructions that make up the program. The user does not need to know what kind of program is being run since MPE XL can determine the mode in which a program should execute.

Introduction to Native Mode

The 900 Series HP 3000 systems have a second environment known as 'Native Mode'. Native Mode 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.

Now that we have a clear understanding of what Compatibility Mode and Native Mode are, we can review the tools that make up the Migration Toolset.

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. All programs and SL's found will then be analyzed by OCA.

The types of potential incompatibilities that can be identified by OCA include:

- References to MPE V Intrinsics 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 that may be required to run a given program.

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 some of the incompatibilities listed in the OCA report are referred to as 'potential incompatibilities'.

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 that contains a list of file names, a list generated by OCA, or from any other source, can be processed 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 called procedures 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 run later 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 Intrinsics 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 are 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 use RTM 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 Planning stage of migration to the 900 Series HP 3000 systems. The Migration Toolset is an orderable product for MPE V/E based 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).

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. Once the MPE V SYSDUMP has been created on an MPE V system, it should be mounted on an MPE XL system. DIRMIG is then run on the MPE XL system and reads the MPE V SYSDUMP tape, extracting the information from the SYSDUMP tape.

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.

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.

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. Cargnoni and I. Janet Garcia, INTEREX 1986 Detroit Conference

MPE V to MPE XL Migration Overview by R. Gregory Stephens, INTEREX 1987 Vienna Conference and INTEREX 1987 Las Vegas 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 Business Unit 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.

Disaster Recovery Planning: A Best Practice at HP

Samuel S. Sudarsanam
Hewlett-Packard Company
Mountain View, California USA

Scenario

At 3:45 P.M. Monday July 14, 1986 during a severe thunderstorm a single engine plane carrying two people was struck by lightning as it flew over the data processing center. The plane was out of control and crashed into the roof and the plane caught on fire. The weight of the plane caused the roof to collapse into the computer room. The computer equipment was crushed and was on fire. Two operators were taken to the hospital in critical condition and six programmers had minor injuries. The data center was closed off and declared off-limits by the fire marshall. All backup tapes and backup computers were 10 miles away from the damaged data center. A Disaster Recovery Plan (DRP) was in place spelling out teams, leaders, call-trees, and procedures. Within two days, the data processing activities were reinstated.

The above mentioned scenario may not sound real but it is true that disasters do occur in a data processing environment causing loss of personnel, assets, and competitive edge.

What is a DRP?

A DRP is a comprehensive document containing the actions required to restore data processing activities in the most timely and effective manner. It is intended to reduce the confusion created as a result of the disaster by clearly defining a course of action.

Management Support of DRP

Generally, a DRP is given a low priority in a data processing environment since people think that the disasters would not occur in their organizations. A strong management support and commitment is needed to develop a good DRP. Management must enforce a policy to have a DRP in place to recover from a disaster. Funds and resources need to be allocated to ensure that this happens. As a rule of thumb, 2% of the DP budget should go to a DRP. Presently in industry only 0.5% of the DP budget is spent on a DRP.

Make-up of a basic DRP

A basic DRP includes:

* An identification of those individuals who would be responsible for disaster recovery activities, as well as an outline of their respective duties in the DRP,

* An explanation of the different levels of "disasters" and a plan of action for each,

* Procedures for continuous backup and storage of vital data to ensure the availability at the backup location until the data center can be renovated or rebuilt,

* Alternate processing methods for maintaining continuity of the business transactions that have been identified as critical for the company and its entities. These methods are planned for interim use following any disaster until regular data processing can be resumed.

Objectives of a DRP

A DRP has the following objectives:

* To reduce confusion during a chaotic period by giving a clearly defined course of action, and thus providing for an orderly and timely recovery from a major interruption of data processing services,

* To identify the personnel, resources, and functions necessary for continued operations in the event of a disaster,

* To identify possible locations that can be used to maintain processing in the event of a disaster,

* To specify the steps necessary to relocate the data center to an alternate site, if required,

* To identify the computer systems that are critical to the operation of the company and define alternate procedures for on-going support in the event of a long-term computer outage,

* To identify computer-related equipment and capacity that would be necessary for temporary replacement processing service and resumption of normal computing service,

* To document the procedures for safeguarding production data and application and system software off-site storage,

* To use an established plan to resume normal processing within predefined limits after a disaster.

HP's Corporate Computing Center's DRP

HP's Corporate Computing Center (CCC) has a DRP to ensure recovery from loss of vital records and processing time. The DRP of HP's CCC includes statement of standards, methodology, disaster recovery outlines for each CCC HP3000, application priority schedules, guidelines for developing disaster recovery procedures for user applications, and equipment off-site agreements. It defines the classification of a disaster based on the intensity of the damage. The following are the types of disasters:

Type 1 - Disaster affecting the entire data center (i.e., earthquake, bombing, etc.)

Type 2 - Machine unavailable for an indefinite amount of time (i.e., fire, water damage, cave-in, etc.)

Type 3 - Machine cannot be repaired within approximately 48 hours (CE assessment on damage)

Type 4 - Machine resources inadequate in accommodating user needs.

HP's CCC has also defined the priority of applications based on their importance. The following are the priority levels:

Priority A - Essential to survival of entire company

Priority B - Essential to the operation of the division

Priority C - New programs in the development stages

Priority D - Training applications

In a DRP various teams are formed to perform specific tasks to recover from the disaster. Each team has a leader to coordinate the tasks and to interface with other teams to effectively recover from the disaster. Disaster control team, production recovery team, non-production team, and restoration team are in action during the recovery time.

Disaster Control Team

In the event of a disaster, this team will get together to assess the course of action and make a decision to implement a DRP. The members of this team photographs the area that was damaged to provide the information to the risk management department. The team also notifies the insurance company(s) and begins the claim processing.

Production Recovery Team

This team will recover the computer systems, all data, and telecommunication lines. The team members will also be responsible for recovery of the production jobs and notification of all users of the status of their applications.

Non-production Recovery Team

This team is primarily responsible for user contact and installation of the hardware/software. The team members will also recover non-production applications and keep in touch with the users.

Restoration Team

This team primarily assesses the damage and reviews the disaster. The team members will be in charge of ordering the equipment and supervising the rebuilding of the facility.

Any event causing loss of computing resources estimated to be more than 24 hours will be considered a disaster that needs to be recovered. Here are the following steps that should be taken:

1. EXECUTE EMERGENCY PROCEDURES	Fire, emergency alarms and other defined procedures are performed.
2. NOTIFY MANAGEMENT	Notification of data center management. Decision to implement a DRP.
3. ASSESS INITIAL DAMAGE	The extent of damage to the data center should be determined.
4. NOTIFY DISASTER CONTROL TEAM LEADER	Disaster control team leader should be informed that a disaster has occurred and a plan is to be implemented.
5. ASSEMBLE ALL TEAMS	All the teams are notified to play their roles actively.
6. ESTABLISH CONTROL CENTER	A control center will be in place to be in touch with the disaster situation.
7. OPEN A DISASTER RECOVERY LOG	Log all the events to inform the management and the users.

****

DRP Testing

Like software, a DRP is tested for accuracy and effectiveness. Testing is to identify any problems in a plan which might prevent the recovery from being successful. A thorough testing of a plan is essential to check that each and every section of a DRP is functional in the time of disaster recovery.

DRP Rehearsal

Periodically a DRP is rehearsed to verify the recovery actions. All the members of the recovery team and users participate in the rehearsal to ensure the recovery of the data processing activities. A DRP may look very extensive and real but unless it is rehearsed, there is no guarantee that it would work in a real disaster.

DRP Maintenance

As an organization grows, its data processing requirements and priorities change. These changes need to be reflected in DRP. Since there are changes in DRP, it needs to be tested again for any shortcomings.

Conclusion

An effective DRP is vital to bring the business activities back to a normal condition in a short period of time. A well-tested and rehearsed plan could save dollars and assets. It may seem tedious and expensive to develop a good DRP but it is necessary to have a plan in place to regain control in an organization. HP firmly believes in a complete and a thoroughly tested DRP. HP also practices DRP by providing continual management support and having a sound plan in place in all of its data centers.

INCREASED PRODUCTIVITY WITH MPE XL MOUNTABLE VOLUMES

Kendall Sutton
Hewlett-Packard
19447 Pruneridge Ave.
Building 47UE
Cupertino, CA. USA 95014

1.0 Introduction

This paper discusses the new Volume Management facility of the MPE XL operating system. MPE XL Volume Management handles disc volume sets, volume classes, and volumes. It creates volumes, mounts volumes, and informs the operating system about volume related data. This paper will cover the specifications and usage of the MPE XL Mountable Volumes facility.

2.0 Specifications

2.2 Volume Management Components

2.2.1 Volumes

In MPE XL a volume is a disc pack, or, in the case of Eagle drives, the disc drive itself. When physically recognized by the system, a volume is in one of the following states:

MASTER - The volume is the master volume of a volume set. Further explanation of this type of volume is in a following section.

MEMBER - The volume belongs to a volume set whose master volume is mounted.

LONER - The volume is a member of a volume set whose master is not currently mounted.

SCRATCH - The volume has been scratched by the user. This type of volume is available for initialization.

UNKNOWN - This is a volume that could not be recognized by the system as any of the previous types. It is also available for initialization.

Each volume that is a member of a volume set has a Free Space Map and a File Label Table. The Free Space Map controls the allocation of free space on the volume. The File Label Table contains file labels and file extent information. When a volume is created, it is given a user defined volume name and a system defined volume ID, which is a unique identifier.

Volume names are 16 characters long. The first character must be an alpha followed by alphanumeric characters. The underbar ('_') is also allowed after the first character. All alpha characters are upshifted.

2.2.2 Volume Sets

A volume set in MPE XL can contain 1 to 255 members or volumes in its life time. Volume sets may contain 0 to 255 volume classes. A volume set must contain a master volume. The master volume contains the volume set configuration in the Volume Set Information Table, and the root directory in addition to the Free Space Map and the File Label Table. When the volume set is created, it is also given a user defined name and a system defined volume set ID.

Volume set names are 32 characters in length. The first character must be an alpha followed by alphanumeric characters. The underbar, '_', and the period, '.', are also allowed after the first character. All alpha characters are upshifted.

The MPE XL volume set name syntax supports the MPE V volume set name syntax. The MPE V volume set name is now a flat name and NOT a hierarchical directory name.

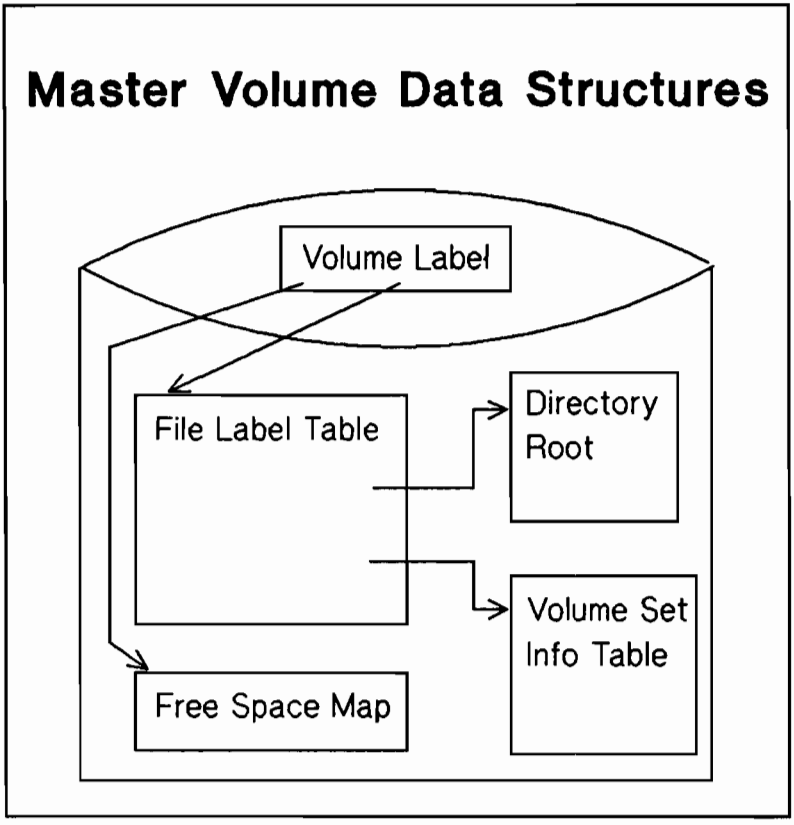
2.2.3 Volume Classes

A volume class is simply a subset of the volumes in a volume set. They may contain 1 to 255 volumes. They are logical entities that control the allocation of disc storage to certain volumes. Volume classes do not need to exist within a volume set; but for compatibility purposes, the volume class DISC should be configured. Volumes may be in one volume class or in several.

Volume class names are 32 characters in length. The first character must be an alpha followed by alphanumeric characters. The underbar, '_', and the period, '.', are also allowed after the first character. All alpha characters are upshifted.

2.2.4 Master Volumes

The master volume is the only volume needed to define a volume set. As stated before, it contains the volume set configuration in the Volume Set Information Table for the volume set of which it is a member. It also contains the root directory for the volume set. See the figure below.



When the master volume is mounted, the volume set is considered mounted. The master volume must be mounted before any file access can be made to other volume set members.

The master volume for the system volume set is special. It contains all the disc bootable images and system configuration. It must be mounted for the system to boot or run.

2.3 Data Availability

MPE XL Volume Management attempts to keep as much data as possible available to the system. This implies that if all the volumes in the volume set are not mounted, the user will be able to access the data from the available volumes in the volume set. MPE XL Volume Management easily allows the user to partition the data so that if a volume goes down, most of the files are still available.

Data partitioning is achieved by restricting where the files are built. There are three levels of data partitioning They are:

- **Volume Set restriction**
- **Volume Class restriction**
- **Volume restriction**

The default volume restriction is the volume set of the group that the file is in. This means that the file extents are placed on any volume within the volume set. Note that a file cannot span volume sets. When a volume goes down, this restriction has the highest probability of stopping access to files. If the master volume goes down, then access to the entire volume set is denied for disc space allocation.

The next level of volume restriction is the volume class. The volume class restriction has to be specified at file creation time. Thus the file is placed only on the volumes within the volume class. If the volume class is a subset of the volume set, then the probability of a disc going down and preventing access to your data has been reduced. If the master volume goes down, then access to the entire volume set is denied for disc space allocation.

The most granular level of volume restriction is the volume. Again the volume restriction has to be specified at file creation time. The file extents are placed only on that volume. With this restriction, the probability of a disc going down and affecting access to data has been reduced further. The master volume must be up for space allocation.

Another type of restriction is the use of mountable volume sets in general. Multiple volume sets partition data very well. Non-system volume sets can easily be moved from MPE XL system to MPE XL system. Another advantage is that disc drives may be shared. For example, your development volume set is placed in the disc drives during the day, and your accounting volume set is placed in at night. Finally back-ups will be able to be done by volume sets. Having multiple volume sets with only a few members is similar to having multiple volume classes on one volume set, except that volume sets can be moved and backed-up separately.

3.0 Advantages of Volume Management

The Volume Management subsystem has several advantages over the Private Volume subsystem. These new features give the user more power and flexibility. These advantages include the following:

1. Consistent syntax across all commands.

2. Don't need the entire volume set mounted to obtain data.

3. Volume set information travels with the volume set.

4. File system perception of all volumes is the same.

3.1 Consistent Command Syntax

The CI commands are consistent in that the same keyword means the same thing from command to command. There are two new keywords, "ONVS" and "HOMEVS". The keyword "ONVS" means that the action of the command is occurring on the volume set specified. When no volume set is specified, the default is the system volume set. The "ONVS" keyword is available on the following commands: NEWACCT, ALTACCT, PURGEACCT, NEWGROUP, ALTGROUP, PURGEGROUP, REPORT, and STORE.

The "HOMEVS" keyword, which is available on the NEWGROUP and ALTGROUP commands, specifies the home volume set for a given group. That is, the volume set on which files in that group will reside. Examples of how these new keywords are used will be presented in a following section.

3.2 Volume Set Mounting

In order to access data from a volume set, only the volumes where the data is located and the master of the volume set need to be mounted. Even the system volume set can be partially mounted.

3.3 Volume Set Information

The information about the volume set travels with the volume set. This information includes the number of volumes in the set, the number and names of the volume classes in the volume set, and the number and names of the volumes in the volume classes. The master volume contains this information.

With this information on the volume set and not in the system directory, it is easier to move a volume set from one system to another. All that is needed is the correct accounts and groups on the system for binding.

3.4 File System Perception

The MPE XL file system views volumes in the same way whether they are members of the system volume set or members of a non-system volume set. Mountable volume sets are not a special case in the file system. Nearly everything one can do to a non-system volume set, one can do to the system volume set. For example, volume classes may now be added to the system on-line. Even system volumes can be added on-line, provided the I/O path has been previously configured.

4.0 Operational Differences

The Volume Management subsystem has some differences from the Private Volume subsystem. The major areas are:

1. Specifications.

2. Directory.

3. Creation and configuration.

4. Commands.

4.1 Specifications

The new Volume Management subsystem provides for larger volume sets and volume classes. The Private Volumes facility allowed only 8 volumes per set or class. As mentioned before the new subsystem allows up to 255 volumes per set or class.

4.2 Directory

In the Private Volume subsystem, the volume set, volume class, and volume information was kept in the directory as a volume set directory node. In MPE XL, creating a volume set does not create directory entries. This information is kept in the Volume Set Information Table on the master volume of the set. There is no permanent directory information kept on the system telling it which volume sets may mounted. As long as the I/O path to the device has been configured, any volume set may be mounted on any system.

In MPE V there are several ways in which to create directory nodes on a private volume. Under MPE XL Volume Management, there is only one way. For example, to create a group node on a volume set, one uses the NEWGROUP command with the new "ONVS" keyword. A complete explanation of how to set up accounting on mountable volume sets appears in a following section.

4.3 Creation and Configuration

In MPE XL, SYSDUMP is not used to create the system volume set. A new subsystem, SYSGEN, is used to configure and create the system volume set. The VINIT utility is no longer used to create volumes, volume classes, or non-system volume sets. It has been replaced with a new utility, VOLUTIL. The User Interface section of this paper will present a step by step guide on using VOLUTIL to create a mountable volume set.

In today's Volume Management subsystem, device classes are not volume classes. Device classes and volume classes are separate entities and are configured separately. This

distinction allows volumes and volume classes to be created on-line, provided that the I/O configuration is present.

4.4 Commands

The following commands have been removed:

NEWVSET, ALTVSET, PURGEVSET, LISTVS

These commands have been deleted because volume sets are no longer directly related to the directory. As stated above, creating a new volume set does not create directory entries. The functionality of the commands has been placed in the new utility VOLUTIL. NEWVSET has been replaced by VOLUTIL's NEWSET command. Likewise, LISTVS is now SHOWSET, and ALTVSET is now ALTVOL.

Since there are no directory entries for mountable volume sets, there is no direct correlation for the PURGEVSET command. Volume sets may be removed from the system anytime using the new VSCLOSE command. Volumes may be scratched and re-initialized using VOLUTIL.

The following new commands have been added:

VSRESERVE, VSRELEASE, VSRESERVESYS, VSRELEASESYS, VSCLOSE, VSOPEN

Refer to the User Interface section for a complete description of the new commands.

5.0 User Interface

The CI user interface has been made consistent for both the system volume set and the non-system volume sets. Defaults for commands and intrinsics are the system volume set. Keywords have been added to commands to invoke a straight-forward meaning and to be consistent across commands. At this time, the MPE V keyword 'VS' has been deleted from all commands.

5.1 Directory Related Commands

The keyword 'ONVS' has been added to the following commands:

NEWACCT, ALTACCT, PURGEACCT

NEWGROUP, ALTGROUP, PURGEGROUP

REPORT, STORE

The 'ONVS' keyword specifies the name of the volume set on which the action of the command is to occur. If the keyword is not specified, the system volume set, MPEXL_SYSTEM_VOLUME_SET is assumed.

The keyword "HOMEVS' has been added to the following commands:

NEWGROUP, ALTGROUP

The 'HOMEVS' keyword specifies the name of the volume set where the files within that group are to be built. The system volume set is assumed if the keyword is not specified.

5.2 Creating A Mountable Volume Set

The VOLUTIL program provides another form of user interface to MPE XL Volume Management. (It replaces the VINIT program on MPE V.) VOLUTIL commands are used to create volume sets, add volumes to sets, add classes to sets, and to display information about volume sets. It is equipped with an excellent help facility which shows syntax and examples for all commands.

Volume sets are created by first creating the master volume. Additional volumes are then added whenever desired. Let's examine the creation of a volume set we will call REPORT_VOL_SET.

We must first determine which devices contain volumes that are suitable for initializing. For this we use the CI command DSTAT. It provides a display like this:

```
 LDEV-TYPE     STATUS       VOLUME          (VOLUME SET - GEN)
-----------  ---------  --------------------------
 1-079350     MASTER      MEMBER1         (MPEXL_SYSTEM_VOLUME_SET-0)
 2-079350     MEMBER      MEMBER2         (MPEXL_SYSTEM_VOLUME_SET-0)
 3-079350     MEMBER      MEMBER3         (MPEXL_SYSTEM_VOLUME_SET-0)
 4-079350     MEMBER      MEMBER4         (MPEXL_SYSTEM_VOLUME_SET-0)
15-079350     UNKNOWN
16-079350     UNKNOWN
```

We see from the display that there are four system volumes and two volumes of type UNKNOWN connected to the system. Refer to the previous explanation of this volume state.

We run the VOLUTIL program by simply typing "VOLUTIL" at the CI prompt. The volutil prompt is then displayed, and we begin with the NEWSET command.

```
NEWSET sname=REPORT_VOL_SET master=MEMBER1 ldev=15
```

We are specifying the set name to be REPORT_VOL_SET, the master volume name to be MEMBER1, and the logical device to be number fifteen. This command creates the volume set by initializing the master volume.

5.3 Creating Groups and Accounts on the Volume Set

The next step is to create the accounts and groups we need on the mountable volume set. In MPE XL there are no directory or accounting commands from within VOLUTIL as there were within VINIT. All directory commands are issued from the CI, which adds significantly to the modularity and consistency of the MPE XL systems. It does, however, require slightly more effort to create the accounting structure.

In order to create files on the mountable volume set, we must create the accounts and groups on BOTH the mountable volume set AND the system volume set. Let's look at what it take to create the file DATA1.SHIPPING.REPORT on the volume set REPORT_VOL_SET.

```
1)  :NEWACCT REPORT,MGR;CAP=xxxxxxxx

2)  :NEWACCT REPORT,MGR;ONVS=REPORT_VOL_SET
```

```
3)  :NEWGROUP SHIPPING.REPORT;CAP=xxxxxxxx;HOMEVS=REPORT_VOL_SET

4)  :NEWGROUP SHIPPING.REPORT;ONVS=REPORT_VOL_SET


5)  :HELLO MGR.REPORT,SHIPPING

6)  :BUILD DATA1
```

Step one is the familiar way to create an account on the system volume set.

Step two creates an account with the same name on the mountable volume set, hence the ONVS parameter.

Step three creates the group entry on the system volume set, but the HOMEVS parameter tells directory services that the files in this group are going to reside on REPORT_VOL_SET instead of the system volume set.

Step four creates the group on the mountable volume set.

In step five we log on to the group on the new volume set and of course the file is built in step six.

5.4 Adding Volumes to a Set

The NEWSET command creates a volume set consisting of one volume. In order to add volumes to the set we use the NEWVOL command from VOLUTIL. Referring to the previous DSTAT display we see that there is another available for initialization on logical device number sixteen.

```
NEWVOL vname=REPORT_VOL_SET:MEMBER2 ldev=16
```

This command specifies that we are creating a new volume belonging to REPORT_VOL_SET that is to be named MEMBER2 and that the volume to be used is residing on ldev number sixteen. Immediately upon completion of this command, the new volume is available for file space allocation. We may enter another DSTAT command now and see that the new volume set is indeed in place:

```
 LDEV-TYPE     STATUS       VOLUME          (VOLUME SET - GEN)
-----------  ---------  --------------------------
 1-079350     MASTER     MEMBER1         (MPEXL_SYSTEM_VOLUME_SET-0)
 2-079350     MEMBER     MEMBER2         (MPEXL_SYSTEM_VOLUME_SET-0)
 3-079350     MEMBER     MEMBER3         (MPEXL_SYSTEM_VOLUME_SET-0)
 4-079350     MEMBER     MEMBER4         (MPEXL_SYSTEM_VOLUME_SET-0)
15-079350     MASTER     MEMBER1         (REPORT_VOL_SET-0)
16-079350     MEMBER     MEMBER2         (REPORT_VOL_SET-0)
```

5.5 Adding A Volume Class to a Set

Some MPE V utilities, such as STORE/RESTORE, that will be used on an MPE XL system, require the volume class DISC to be present on the volume set. In order to add the class DISC we use the following VOLUTIL command:

```
NEWCLASS cname=REPORT_VOL_SET:DISC volumes=MEMBER1,MEMBER2
```

This command specifies that the class DISC is to be added to the volume set REPORT_VOL_SET and that the volumes named MEMBER1 and MEMBER2 are do be added to the class DISC.

5.6 Backing Up a Volume Set

Performing tape backup on a volume set is quite easy on the MPE XL system. A new option has been added to the CI store command:

```
STORE @.@.@;ONVS=REPORT_VOL_SET
```

This command will store to tape all files residing on the volume set, and only those files on the volume set.

5.7 Other Volutil Commands

Many other commands exist within VOLUTIL. Some of them are mentioned briefly below:

SHOWSET - Displays a variety of information about a volume set.

COPYVOL - Allows the copying of individual volumes.

COPYSET - Allows the copying of an entire volume set. Additionally, it enforces the integrity of the target and source volume sets to prevent inter-mixing of original and copied volumes.

EXPANDCLASS - Adds volumes to an existing volume class.

SHOWCLASS - Displays a variety of information about a volume class.

SHOWVOL - Displays a variety of information about a specific volume.

5.8 Operations Related Functions

The following MPE V commands are supported:

MOUNT, DISMOUNT, LMOUNT, LDISMOUNT

VSUSER, DSTAT, VMOUNT

The MOUNT, DISMOUNT, LMOUNT, and LDISMOUNT commands will support *only* the MPE V volume set name syntax. However, volume class mounts will not be supported. The VSUSER, DSTAT, and VMOUNT commands will be supported the same as they are on MPE V except that volume names are now 16 characters long instead of 8, and volume set names are 32 characters long instead of 27.

The following new commands have been added:

VSRESERVE, VSRELEASE, VSRESERVESYS, VSRELEASESYS

VSCLOSE, VSOPEN

5.8.1 Differences in Operations Related Commands

As mentioned previously, MPE XL Volume Management makes volume sets available to users as soon as they are physically recognized by the system. This means that, by default, the volume set is available to users. In MPE XL, no explicit MOUNT command need be issued in order to access the volume set. The function of this command is now to simply notify the system that the user wants the volume set to remain physically mounted for a period of time.

It is for this reason that we have introduced the new VSRESERVE and VSRELEASE commands. They more properly describe the function, and their use is greatly encouraged over the older private volume commands. Each is explained in detail below.

5.8.2 New Operations Related Commands

The VSRESERVE commands reserves the volume set between file opens for the user. That is, it notifies the system that even when the user does not explicitly have a file open on the volume set, further access will be forthcoming. This prevents the operator from taking the volume set off line. (See the VSCLOSE command below.) VSRESERVE supports the MPE XL volume set name syntax, while MOUNT only supports the MPE V volume set name syntax.

The syntax is:

```
VSRESERVE [MPE XL volset name] [;GEN=num]
```

If no volume set is specified, then the request is for the home volume set of the user's logon group and account. Otherwise the user must specify the full volume set name. The reservation of the volume set automatically ends when the user logs off. The generation number is the same as MPE V's.

The VSRELEASE command negates a previous VSRESERVE command. The syntax is:

```
VSRELEASE [MPE XL volset name]
```

If the volume set is not specified, then the home volume set of the user's logon group and account is used.

The VSRESERVESYS command reserves the volume set for the entire system. This command indicates to the system that the volume set is to remain on-line until an explicit VSRELEASESYS command is issued. Unlike the VSRESERVE command, this command is unaffected by logging off.

The syntax is:

```
VSRESERVESYS [MPE XL volset name] [;GEN=num]
```

The volume set name must be specified.

The VSRELEASESYS commands negates the previous VSRESERVESYS command. It indicates that the system wide reservation of the volume set is no longer in effect. It has no effect on VSRESERVE commands issued by individual users on the system. The syntax is:

```
VSRELEASESYS [MPE XL volset name]
```

The volume set name must be specified.

The VSCLOSE command designates to the operating system that the volume set is going to be removed. This command replaces the MPE V DOWN command. This command will restrict access to the volume set. Any job/session that 1) has NOT done an explicit VSRESERVE/MOUNT on the volume set and 2) currently has NO files open on the volume set, will be denied access to the volume set. The syntax is:

```
VSCLOSE [MPE XL volset name] [;NOW]
```

The volume set name must be specified. The VSCLOSE commands patiently waits until all the files have been closed on the volume set unless the 'NOW' keyword is specified. If the 'NOW' keyword is specified, all the users of the volume set will be aborted and the volume set will be ready for removal.

The VSOPEN command opens a previously closed volume set. Its syntax is:

```
VSOPEN [MPE XL volset name]
```

The volume set name must be specified. After the VSOPEN command is issued, the volume set is ready for use. Recall that the default for a volume set is for it to be available. Therefore a volume set is considered open until it is explicitly closed.

6.0 Practical Advantages of Volume Management

6.1 System Volume Set

One big advantage of MPE XL Volume Management, in terms of the system volume set, is the ability to conveniently add a new disc drive in critical situations. If, in the middle of a large application, it was determined that there was not enough spool file space, or not enough permanent disc space, it would not be necessary to bring the system down in order to add a new drive. Assuming that the I/O path had been previously configured, the new drive could be added to the system, on line, *while the application was running*!

6.2 Mountable Volume Sets

There many reasons to use one or more mountable volume sets along with the system volume set. Many of the advantages are not limited to MPE XL, but apply equally well to MPE V private volumes.

6.2.1 Lessen Down Time by Reducing Reloads

Let's assume that all the volumes on a four volume system are members of the system volume set. Any serious hardware or software failure on any of the drives could force a reload of the entire system. The result is that no users would have access to the system for some time.

Now consider the a similar system consisting of a two volume system volume set and two mountable volume sets. Now the chances of a catastrophic disc failure forcing one to bring the system down for a reload have been cut in half. If a system reload is necessary, it won't take as long since the number of system files has been reduced.

If the catastrophe occurs on one of the other volumes sets, then only that volume set will need to be reloaded. Users that are not dependent on that set will continue to have full access to the machine during the operations that are necessary on the failed volume set.

6.2.2 Avoid Down Time During Hardware Only Failures

In the case of hardware failures that do not affect the data on removable disc packs, MPE XL allows the media to be moved to another drive. This can be done in most cases *without bringing the system down.*

If the failure occurs on a mountable volume set, then access to files that have extents on the failed volume will be affected. All other file access can proceed normally. The pack could be moved to another disc drive and that drive can be connected in place of the failed drive. (The new drive must be set to the same ldev number of the previous drive.) The powerfail characteristics of MPE XL will allow the new drive to be recognized as a remount after powerfail, and access will then proceed normally.

Even system volumes can be moved from one drive to another (again, they must be the same ldev). Access to the system volume set will be limited more severely during the swap, however, because the failed volume may contain directory information for files residing anywhere on the volume set.

6.2.3 Increased Utilization of Limited Hardware Resources

By grouping applications and data in mountable volume sets, a limited number of removable media disc drives can be more fully exploited. Volume sets containing applications needed during the day can be mounted each morning. Large batch jobs that can run overnight could reside on other volume sets which can be mounted in the same drives in the evening.

6.2.4 Critical Applications can be Quickly Moved

In the case of a serious CPU failure or problem with the system volume set, a critical application could be moved to another machine. This, of course, would take some preparation. The application and data would be restricted to one or more mountable volume sets. The proper accounting structure would need to be put in place on the new machine. Once this is done, the drives could be connected to the new system (again, assuming the I/O path was previously configured) and they would be recognized automatically. Optionally, a volume set already on the new system could be removed and replaced by the new set.

7.0 Summary

MPE XL Volume Management has made volume management consistent for both system and non-system volume sets. The user sees that the only difference between system and non-system volume sets is the fact that the system volume set is needed to run the system. The operator now has control over the right granularity, the volume set. Finally, MPE XL Volume Management is more robust than its predecessor. Not all the volumes in the volume set have to be mounted before the user is able to access files and volumes may be added to system or non-system volume sets while the machine is up.

Acknowledgments

I would like to acknowledge Rick Ehrhart, the original designer of MPE XL Volume Management, for his many contributions to this paper. It is because of his talent for thoughtful and careful design that Volume Management is the outstanding and flexible subsystem that it is today. I would also like to thank Walt McCullough for assistance in directory related issues, Howard Burrows, who coded VOLUTIL, and Gary O'Neall for his tremendous support.

TRENDS IN IMAGE

C. Bradley Tashenberg
Bradmark Computer Systems
4265 San Felipe, Suite 820
Houston, TX 77027

TRENDS IN IMAGE

C. Bradley Tashenberg
Bradmark Computer Systems
4265 San Felipe, Suite 820
Houston, TX 77027

Isn't it incredible how hindsight can be so clear, and yet looking only a few years into the future can make you feel legally blind? For sure, this is what HP (and many other minicomputer vendors) must have felt and experienced during the early to mid 1970's.

Isolating the years from 1976 to 1978, what occured was a dramatic shift in technology. The original one-board systems which were limited to 64 KB in memory were quickly obsoleted by the introduction of the semiconductor. Systems like the HP3000 made a quantum leap forward in technological advancement from the core-based CX model to the semiconductor-based Series II.

Proportionally, the processing power of the HP3000 increased from a 64 KB, 5-terminal system with limited disk capacity of half a megabyte to a 16-terminal system with substantial disk capacity.

Within 18 months of the announcement of the Series II, yet another HP3000 was on the market (the Series III), offering even broader functionallity and greater processing power over its already successful predecessor.

The trend in ever-continuing advancements in the HP3000 systems line has gone on for a full decade, until the meaning of the term minicomputer totally loses its significance.

Who would have thought that over the course of a decade, the HP3000 would be a contender to the mainframe, supporting up to 350 terminals and capable of housing 16 megabytes of main memory. That doesn't sound like a minicomputer to me.

And yet, it pretty much epitomizes the spiralling growth that HP experienced throughout the late 70's and early 80's. The end result is that nobody would ever consider the CX even a distant relative to the 70, and yet it is.

The advances made in HP hardware provide better throughput and increased performance to handle larger loads, and still this doesn't seem to be enough to meet the needs of the high-end users who are anxiously awaiting the Spectrum

family of machines.

HP has made some very significant advancements in the HP3000 and its peripheral hardware, but an interesting question arises: what effect has this kind of growth had on software? Has the software matured with the hardware in providing needed capabilities and processing power to handle today's demanding DP tasks?

Unquestionably, one of the reasons for the HP3000's success has been the IMAGE/3000 database management system. IMAGE is unusual in that it is provided free of charge as a standard for application systems on the HP3000. Most systems are written with IMAGE at their core, with the knowledge that these applications can be transported to any other HP3000 and run without modification, keeping in line with HP's commitment to compatibility.

One might think, then, with IMAGE's wide use and great success that HP would have made efforts throughout the years to keep it as functional and state-of-the-art as its hardware; but surprisingly, up until the release of TurboIMAGE last year, IMAGE had gone almost unchanged from its humble beginnings on the lowly Series II.

But perhaps even more suprising is that HP has relied on its IMAGE utility suite--the DBUNLOAD and DBLOAD programs--for years, even though these programs were barely adequate back in the early years and are totally inadequate for today's large IMAGE and TurboIMAGE databases.

Most users' requirements have focused on the need for additional throughput from their databases. HP did take a positive step forward with TurboIMAGE, but many users who have upgraded to Turbo have been disappointed to find only minimal or no performance improvement. Some sites have even reported degraded performance after installing TurboIMAGE. The rationale seems to be that HP misunderstood the true performance problems in IMAGE and miscalculated the improvements it would provide.

In IMAGE, all data being accessed is moved into buffers in the Database Control Block (DBCB). Because IMAGE only processes one intrinsic call at a time (the meaning of the term "single-threaded"), contention for the DBCB would result. HP determined that alleviating this bottleneck would substantially improve performance in TurboIMAGE. So HP changed things around in Turbo so that many Database User control blocks (DBUs) would be created to concurrently process transactions which could complete in one I/O (DBGET, DBUPDATE, etc.). This is refered to as "multi-threading."

Unfortunately, most of the performance problems in IMAGE are

not caused by DBCB contention, rather, they result because the disk drives cannot keep up with the system. This is the fundamental bottleneck on the HP3000, and this is why HP caches both system and disk memory--to reduce the number of disk accesses by storing frequently-accessed data in cache.

But this is hardly enough to adequately improve IMAGE and TurboIMAGE performance. The most effective methods of doing so are to keep the database in tune so it requires less disk I/O's to accomplish the work it must do. This is the nature of our recommendations for optimizing performance in IMAGE and TurboIMAGE databases.

One of the factors which can degrade the performance of an IMAGE data base is the presence of synonyms in master data sets. Synonyms (or secondaries) are a by-product of the very efficient method of hashing which IMAGE uses when adding entries into, and retrieving them from, master data sets.

IMAGE was designed to retrieve an entry in a master data set with one I/O. This is done by hashing the value of the key item of the incoming entry into an address in the master data set and then storing the entry at that location. To later access this entry, one supplies the same key value, and IMAGE hashes it and looks for the entry at that address. This method normally provides very quick retrieval.

Because of the nature of hashing, it is improbable that entries will always hash to unique addresses, and IMAGE's hashing method is no exception, with an average unique-hit rate of about 70% (for a master data set which is 80% full). IMAGE takes an entry which hashes to an address already occupied by an entry (called the primary), places it elsewhere in the data set, and chains it to the primary via a doubly linked list (the synonym chain). IMAGE attempts to place the secondary entry in the same block as the primary; if it cannot, it will try to place it in an adjacent block.

Now, still additional entries may hash to the same primary address and they, too, must be relocated. But IMAGE does not simply append them to the end of the synonym chain: it reads each and every record on the chain because it must make sure that an entry with the same key value as the entry being added does not already exist in the data set; if it does not, IMAGE will add this entry to the end of the chain.

The same synonym chase is performed when retrieving these secondary entries using mode-7 (keyed) DBGETs or DBFINDs (which look for the master entry by key) because they, too, will start at the primary entry and read down the synonym chain to find the record being sought. If the synonym is located in the same block as the previously-read record,

IMAGE does not have to do another disk I/O to read it; otherwise, it must perform another I/O. This means that any intrinsic call which tries to DBPUT a new master entry or DBGET an existing master entry in keyed access mode can potentially take more than one I/O.

A condition called clustering, in which many entries hash to the same locations, can require that even more I/Os be done to DBPUT new entries because IMAGE may have to read many blocks before it finds a free location. A DBGET of this entry would, however, require less I/Os because IMAGE is able to follow the synonym chain pointers and does not have to read serially through blocks. In order to determine whether or not the potential performance problem of synonyms is actually degrading performance in a master data set, it is necessary to have information not only about the number of secondaries in the set, but more importantly the distribution of the entries in the data set.

Use a diagnostic to analyze your master datasets. The first indication of a potential performance problem is if the primary/secondary ratio exceeds IMAGE's norm. Another indication of a problem is if there are many synonym chains of lengths greater than two or three--chains of such lengths require an inordinate number of I/Os. The master set should contain a relatively uniform distribution of entries; unusually saturated areas indicate clustering.

If the diagnostic reveals that there is an inordinate number of secondaries or that there are incidents of clustering, this does not necessarily mean that performance is suffering as a result. After all, how many additional I/Os does it take for IMAGE to handle synonyms? Sometimes one or two or three--sometimes hundreds! But very often synonyms impose virtually no additional I/Os. It is important to realize that while synonyms are potentially a performance problem, the extent of degradation is directly associated with the data set's blocking factor. Remember that IMAGE will try to PUT synonyms in the same blocks as their primaries, which enables them to be addressed without doing any additional I/Os than if they were primaries.

So if you have a master data set, for example, with a very high percentage of secondaries (say 50%) and a blocking factor of 20 but an even distribution of entries across the set, don't worry about it--chances are IMAGE is able to place the secondaries in the same blocks as the primaries and therefore does not need to perform any additional I/Os to PUT them or GET them. If, on the other hand, you have a master data set which has a very long record length and is therefore blocked at one and you have only 20% secondaries, you would be wise to try to reduce the percentage of secondaries in that master.

If the diagnostic reveals that synonyms are a problem and the data set has a blocking factor so low that the effect of these synonyms cannot be absorbed and require many additional I/Os, something should be done to reduce the number of synonyms and to better the overall distribution in the data set. But how can we lower the number of secondaries and improve the distribution when they are solely a result of IMAGE's hashing function, which is etched in stone and beyond our control?

The method is quite simple: change the capacity. Because the capacity is an element in IMAGE's hashing algorithm (it is used as a divisor or a modulo, depending on whether the search item data type is alphanumeric or numeric), it is extremely influencial in determining master dataset performance. An efficient capacity yields more primaries and less secondaries, and an even distribution of entries throughout the set.

The trick here is in determining an efficient capacity.

It is by now gospel that the method for choosing a capacity for a master dataset is to determine how much free space will be needed to allow for growth, increase it if this does not leave 25% free space, and then choose the nearest prime number. The theory here is that prime numbers tend to create uniqueness and therefore primaries; and the more free space IMAGE has to work with, the better chance that it will find empty locations in which to place (primary) entries. Some even suggest setting master data set capacities to double or triple the active entry count.

The latter theory has been disproven by many. There is a threshold of free space which is of benefit--anything beyond that does not help, but only takes up more space on disc, more tape and more hours each time the base is backed up, and more I/Os for programs which read the set serially. Twenty to 25% percent free space is about right.

But the prime number theory does not seem to hold true either. Our studies indicate that a prime capacity can yield the worst results when compared with other capacities it its vicinity. (These are results of tests using alphanumeric keys of different lengths on several application systems, large and small.) Similar studies have been documented by others, including HP personnel.

The test results demonstrate not only that prime capacities often do more harm than good but that a master data set's efficiency is extremely sensitive to its capacity and that the primary/secondary ratio may vary a great deal even within a very narrow capacity range--even if the capacity is increased or decreased by only one. This suggests that

extra effort should be taken to determine an efficient capacity.

We have not found an algorithm to determine efficient dataset capacities, so we conduct an empirical study to determine the best capacity for a master dataset. This study is done by extracting a percentage of the entries from a specified master dataset and sampling various capacities using these entries, each time incrementing the capacity by one. During our sampling, we accumulate the statistics of primaries-to-secondaries and distribution factors for each capacity sampled.

Once you have determined an efficient capacity for a dataset, leave it alone! A good capacity will continue to do well until the set is very near full, while a bad capacity will always produce a high percentage of secondaries.

So don't just automatically change a master capacity when it hits 80%--you could be sitting on a capacity which gives you very good performance so milk it for as long as you can, right up to near-saturation. On the other hand, if you've got a high percentage of secondaries due to clustering (a situation in which many entries hash to the same locations) and are at only 50% of capacity, it's time to make a change to better the distribution.

It is permanently engraved in the minds of most HP3000 users that the use of integer data types for search items in master datasets is strictly taboo. Most have heard horror stories about DBPUTs to integer-keyed datasets taking several seconds to several minutes. But as is generally true, lack of knowledge breeds fear. Used properly, integer-keyed datasets outperform and are more disk-efficient than those with alphanumeric keys.

There is one very simple rule for successful use of integer keys: do not permit the key value to exceed the capacity. Period. If this rule is followed, you will never have a problem. IMAGE has two algorithms which are used for placing entries into master datasets: one for alphanumeric keys (data types X, U, P, and Z) and one for integer keys (data types I, J, K, and R).

The alphanumeric hash is a bit hash which breaks the key value into bits, mixes them up, and then hashes the key.

The numeric algorithm simply takes the key value, substracts 1, divides it by the capacity, and adds 1 to the remainder. The result is that the entry is placed at the random record location equal to its key value. (Due to a bug in the algorithm, if the capacity of a dataset exceeds 65,535, the entry is placed at the random record location equal to its

key value plus one.)

For example, a key value of "10" would be placed at random record number 10 (or 11 if the capacity is greater than 65,535). Since every key value in a master dataset must be unique, there is always a free location available for each incoming entry. And since each entry has calculated for it a discreet record number, there are never any secondaries.

If the keys are sequential values with no gaps, there exists a solid block of entries with no free locations at all between them. This is perfectly efficient; but this is also the downfall of the dataset when its key values exceed the capacity: the classic problem with integer keys is that old values are deleted opening up free space for incoming entries. At some point, a key value exceeds the capacity and incoming entries begin to hash to already-occupied locations.

An integer-keyed dataset whose capacity exceeds the highest key value has the fastest access of any master dataset because there are no secondaries. Therefore, there are no wasted I/Os in addressing secondaries.

There can also be significant disk savings to using integer keys. An integer can be represented in less disk space than an alphanumeric key. For example, an I2 takes only one word of disk, while its alphanumeric equivalent--an X12--requires six words. Even more disk space may be saved because it is not necessary to leave any free space in the dataset for hashing (alphanumerically-keyed master datasets should have 20 - 25% free space to help hash primaries).

If all this sounds too good to be true and you're wondering why everyone isn't using integer keys everywhere, it is because there are not too many practical applications for integer keys. The most appropriate use is for a master dataset whose key value is incremented by one each time and whose entries are never deleted. Perhaps the most widely-used example of this is control codes.

It is very important to remember that significant performance problems can result if a key value exceeds the dataset capacity. This is because of the way in which IMAGE handles synonyms--entries which hash to already-occupied locations. IMAGE's method for placing a synonym is to read serially and place the incoming entry at the first free location.

This is a real problem for integer-keyed datasets, because there is likely to be nothing but solid entries and several hundred or thousand I/Os may be necessary before the entry can be DBPUT. The normal indication of poor performance in an integer-keyed master dataset is that DBPUTs take an

inordinate amount of time because they are reading through many blocks of to find a vacant location. Normally, however, DBGETs of entries in the dataset are quite fast because they are accomplished via a directed read to the address determined by reading the forward pointer on the primary entry of the synonym chain.

It is easy to misdiagnose a performance problem in an integer-keyed master dataset because most diagnostics report only the primary/secondary ratio, which is likely to be very small. It is not at all unusual to have a horrendous performance problem in an integer-keyed master dataset which has an average secondary chain length of one. Again, the problem is not the speed of the GETs but the speed of the PUTs.

So if you have a master dataset with an integer key, this does not automatically mean you are going to suffer bad performance, but beware of the difficulties if the key values exceed the capacity.

Unlike master datasets, which normally perform well unless they are shy of free space to benefit hashing, are the victims of poor capacities, or are the result of misused integer keys, detail dataset performance degrades from day one and continues to degrade every day due to normal access.

This example illustrates why:

If a data entry clerk inputs an order comprised of ten line items into a detail data set, and that detail data set has a blocking factor of 8, the first eight entries of the order would be located in the first block of the set and the last two entries would be in the second block, and you could retrieve this order with two I/Os. But you did not buy your HP3000 to support only one user. Add several more users entering orders and you may find the ten lines of that order spread out over ten blocks. To assemble that order would take five times as many I/Os.

This is the everyday life of a detail dataset where, because of nothing more than normal concurrent DBPUTs, the chained access to the dataset may not be highly inefficient.

The situation is worsened when entries are deleted from a detail dataset. IMAGE's scheme for reusing space which is made available in a detail set by deleted entries is to chain all these locations together (by writing a pointer into the first word of the record which links it to the previously deleted entry) on a singly-linked list called the delete chain. IMAGE reuses these deleted entries before once again appending entries after the set's highwater mark.

Because the entries being deleted are likely scattered throughout the set, any incoming entries which take their locations are also dispersed. The incoming entries which make up our 10-line order would therefore not only be located in different blocks, but these blocks would likely be spread widely throughout the dataset. So not only does IMAGE have to do extra I/Os to read down that detail chain, it also requires more time to move the disk head around to read those blocks. The more deletions, the worse the situation gets.

It is due to these fundamental processing inefficiencies that HP recommends that databases be unloaded and reloaded from time to time to improve performance. This gets rid of the deleted entries and, more significantly, a chained DBUNLOAD/DBLOAD puts the entries back in the dataset so that the entries are contiguous on their respective chains. This can mean a very substantial performance improvement because the entries can be retrieved in many less I/Os.

But a DBUNLOAD/DBLOAD has several drawbacks, the most immediately apparent one being that it requires a human to swap tapes for several hours or days. Another not-so-apparent drawback is that in addressing performance problems a lot of the time in the DBUNLOAD/DBLOAD is wasted in the master datasets, which do not benefit at all from being unloaded and reloaded. After all, entries hash into a master dataset based on its capacity and if the capacity doesn't change, everything hashes back to the very same locations where they were before. Much pain, no gain.

So the unload/reload effort should be concentrated on detail datasets only. And it should be concentrated on only those detail datasets which would benefit significantly from being unloaded/reloaded. The important factor here is the fragmentation of the chains in the detail datasets. This can be determined by running a chain diagnostic on detail datasets and looking at the inefficiencies and fragmentation of the chains along the path which is designated as the primary path.

Those sets which reveal significant fragmentation can require many more I/Os than necessary to process those entries--sometimes up to ten times as much, or more! For these datasets, a detail reorganization should be performed to reorder the entries on their chains.

Before performing a reorganization, however, you should make sure the correct path is defined as the primary path. It is crucial that you do this because the reorganization will reorder everything by the chains along the primary path, so you should make the primary path the one with the most chained access. The emphasis here is on CHAINED because

many users miss this critical point and assign the primary path to the path which has the most access, although it may not be chained access.

For example, a detail dataset whose entries have a one-to-one relationship with an associated master dataset (one detail entry per master entry) should never have the primary path assigned to it because the detail contains no chains. This database design flaw is seen time and time again. Quite often, it is the common database construct in which a detail dataset contains a header record for each invoice, and the detail set is pathed to two masters: one is an automatic master which contains invoice numbers, the other is a manual master which contains customer information; and the primary path is assigned to the automatic master. Again, there are no chains snaking through the detail dataset for the invoice number path because there is never more than one header per invoice; however, each customer may have tens or hundreds of invoices. So the primary path should be assigned to the second path--even though the access along the first path may be much more frequent.

If two paths are accessed chained with about the same frequency, you should make the primary path the one with the longest average chain length. A database restructuring tool can accomplish this in seconds, or it can be done via a DBUNLOAD/DBLOAD.

The most recent major trend in IMAGE has been the upgrade to TurboIMAGE. Turbo has increased limits of the number of items and sets that a database may contain, and increased dataset capacities. This permits users to design the larger databases that today's environment requires.

The discussion of TurboIMAGE leads us into the topics of blocking factors, block sizes, and BLOCKMAX, as well as another program that hasn't changed with the times, DBSCHEMA.

One of the most significant enhancement in TurboIMAGE is the support of detail chains containing more than about 65,000 entries. This limitation exists in IMAGE because the entry count for each chain is maintained in a one-word field, and this limits the value to the high value for an unsigned single integer: 65,535. In TurboIMAGE, the chain count is increased to two words--a signed double integer--with a high limit of about 2 billion.

This requires increasing all the master data entries (which comprise the chain heads) by one word for each path. To accomplish this, and to convert the radically-changed root file, HP provides the DBCONV program. This program converts the root file and then all the master datasets except stand-

alone masters to TurboIMAGE format. This process is amazingly fast and painless.

But we all know by now, all too often in data processing, things which are quick are also dirty, and DBCONV is no exception. The conversion to TurboIMAGE can cause inefficiencies which waste excessive amounts of disk space and degrade performance. The investigation and correction of these problems will very likely identify related, long-standing database inefficiencies which were not caused by the TurboIMAGE conversion but which have the same symptoms.

After conversion to TurboIMAGE, your database may need substantially more disk space that it currently uses. A few sectors of additional disk space are used by the root file; but of far greater concern is the space required if some of the master dataset block sizes must be increased.

If a master dataset has enough free space in its existing block to accommodate the extra words, no additional disk space is required. It is likely that most of the master datasets will have the free space available: for these datasets, the block size does not need to be increased.

The amount of free space per block in a master dataset is dependent on how closely divisible the media entry is into the block size. For example, a media entry length of 50 in an IMAGE block size of 512 words is a good fit: (50 * 10) + 1 = 501. Eleven words per block are wasted. (The 1 added in this example represents one word for the bit map which resides at the beginning of each block in a dataset; the length of the bit map is dependent on the blocking factor.) The media entry length of 50 is broken down as 35 words for the data entry including the key, 5 words for the synonym chain, and 5 words for each path.

If there is not enough residual space in the block to fit the extra word or words required by Turbo, the dataset is made bigger. DBCONV converts the database by creating a new file for each master dataset and copying the blocks from each old file to each new one, inserting the extra words required. Datasets which must be increased in size are built one or more sectors larger than previously.

And herein lies a significant problem. A sector (128 words) is normally much more space than is needed--usually, only a few words are required. But a full sector is used because this is the smallest unit of disk that may be allocated for a dataset disk block.

The number of additional sectors required is dependent on the number of paths from the master dataset and its blocking factor. Master sets with several paths and/or high blocking

factors may require upwards of four additional sectors for expansion.

One can hardly fault TurboIMAGE for requiring more disk space--after all, if it provides an increased chain limit it's got to keep track of it somewhere and have someplace to put it. But one can, however, rightfully question DBCONV's method of converting datasets because it unjustifiably wastes disk space.

This illustrates the point:

Let's use the same master dataset described above as an example. It is a manual master with with 250,000 entries which is pathed to two detail datasets. Its block size is 512 words and its blocking factor is 10. If we convert it to TurboIMAGE, how much additional disk space will it require?

The easiest way to find out is to run DBCONV,VERIFY against its database--this will report the amount of additional disk space required by each dataset and the entire database, as well as the amount of disk space required temporarily for the conversion (equal to the largest dataset which requires conversion). It is even possible to run DBCONV on an IMAGE system to see if enough disk space exists to convert to Turbo.

But since we are not all at our terminals and don't necessarily have such a database, here's the math: 35 words for the data entry + 5 words for the synonym chain + 6 words for each path = 52 media entry length. The media entry length has been increased by two words (one for each path).

We cannot use the existing block size/blocking factor combination because 52 x 10 (blocking factor) + 1 word bit map = 521 required space, which is greater than the current block size of 512 words.

So DBCONV will increase the block size of the dataset by one sector (128 words) to 640 words to accommodate the 9 words of overflow. The remaining 119 words in the last sector are empty.

This block size increase has a major impact on the disk utilitation of the dataset. The disk space required by the dataset under IMAGE was 100,000 sectors (250,000 capacity / blocking factor 10 * 4 sectors per block). The conversion to TurboIMAGE refigured the disk usage to 125,000 sectors (25,000 blocks * 5 sectors per block).

The total increase in disk utilization for this dataset is 25,000 sectors. And because 119 words in each block are

empty, 93% of the newly-acquired disk space is unused.

So what is the alternative? A better method of converting this database to TurboIMAGE would have been to preserve the existing block size and reduce the blocking factor from 10 to 9. (This may not be possible if the blocking factor is very low.)

Since the media entry length of 52 x 9 (the reduced blocking factor) + 1 word for the bit map = 469 required space and the block size is 512, there would be 43 words wasted in each block.

By this method, only 43 words per block are unused. But this method requires quite a lot of work because many of the pointers within the dataset must be restructured. The format of the pointers from a primary master entry to its secondaries (the synonym chain) is block number plus position within block, so all the synonym chain pointers must be updated to reflect the new locations. This can be a time-consuming operation.

But this does result in efficient disk utilization and greater blocking effectiveness and maintains the old block size of the datasets. This is also important for another reason: increasing the block size of a dataset results in an increase in memory utilization; and not just for that dataset but for the entire database.

Under TurboIMAGE, blocks of data are read from disk into buffers and these buffers are moved in and out of the DBB (Data Base Buffers) control block. All buffers are shared by all the datasets, so the buffers must be sized to be as large as the largest block size in the database--even if just one dataset uses that large a block size.

The DBB is memory-resident, so a large DBB can cause a program accessing the database to run slower because it requires more memory. In addition, because the DBB has a limit of 32 K regardless of the block size and because the buffer lengths are as large as the largest block size, there is less room for buffers therefore less buffers may be allocated.

The formula for calculating the DBB size is:

$$4000 + 26s + n(b+13)$$

where

s is the number of DATA SETS
n is the number of buffers
b is the BUFFER LENGTH

For a database with 65 datasets, this can mean an additional 8 K of memory to support the enlarged DBB control block and more limited buffer space.

What's more, most of this additional space in the DBB is unused because blocks from datasets which were not increased in size by DBCONV (all the detail sets, stand-alone master sets, and masters with enough residual space) are read into buffers which are too large for them. This means lots of wasted space in the DBB.

So an IMAGE database which is converted to TurboIMAGE can require more disk space to support its master datasets and more memory to support its DBB control block. Again, this is all dependent on the amount of residual space per block in the dataset prior to conversion, so it is a sad truth that the databases which end up being grossly inefficient (many master sets increased) under TurboIMAGE are the ones which were the most efficient under IMAGE because the most efficiently-blocked datasets are those which have little or no free space in their blocks and therefore no room for expansion.

But an increase in resources alone is not the problem to be concerned with because it is unavoidable. The real problems are that the increase is probably excessive, and we've not gotten the proper value for the newly-acquired resources.

Let's go back to our converted master dataset example. The increase of the block size from 512 to 640 left 119 words unused in each block. Now, because the record length is 52 words, we can fit another two records per block, because the media entry length of 52 x 12 (increased blocking factor) + 1 word for the bit map = 625 words.

This increases the blocking factor from 10 to 12 and wastes 15 words in the block. Because more records can be fit into each block, the number of blocks required is reduced and therefore the total disk utilization of the dataset is reduced by about 20,835 sectors.

If this is so, why doesn't DBCONV better optimize disk storage by increasing the blocking factor to 10 to 12 at the same time it is increasing the block size? Because blocking factor changes require pointer adjustments for all the synonym chains and DBCONV doesn't do this, remember?

A solution to the other problem--better utilizing the available buffer space in the DBB--is to standardize all the block sizes in the database, either by reducing the DBCONV-increased block sizes or by increasing the block sizes in the unaffected datasets. What is important is that they should all be the same.

Now, it may come as a surprise that this problem of inconsistent block sizes--made apparent by DBCONV--is nothing new. This problem has probably been lurking in your database forever. It was not caused by DBCONV, but its symptoms and remedy are similar, so it is appropriate to discuss it here. It is the "other side of the coin" to datasets whose block sizes are too large--those which are too small.

If you do a :LISTF,2 of a database, you may notice that several different record lengths occur throughout the files. For example, if the BLOCKMAX (maximum block size) for the database is 512, you will probably see one more more dataset files with record lengths of 640 or greater (the ones increased by DBCONV) but you may also find some datasets sized below 512, at 384 or 256 words.

This is because when the database was initially built, DBSCHEMA, the schema processor, decided what block sizes and blocking factors to assign to the datasets; and in doing so, it attempted to optimize disk utilization rather than I/O. DBSCHEMA was written to choose a block size less than the BLOCKMAX if this resulted in less wasted disk space at the end of each block. This is probably because the program was written in the early 70's when disk space was scarce and expensive. Unfortunately, DBSCHEMA still tries to optimize disk even under TurboIMAGE.

There are two problems with this: one is that because the block size is less than it could be, the blocking factor is also less than it could be. While this may result in high blocking efficiency with little wasted space in each block, the amount of residual space that would be wasted were the block size increased to the BLOCKMAX would be minimal. A lower-than-BLOCKMAX blocksize can seriously compromise the blocking factor, which is foolish because the blocking factor is fundamental to the performance of the database. It is far better to waste a little disk space in favor of a higher blocking factor which will get more records with each disk I/O and therefore minimize disk accesses. Because most HP3000 application systems are I/O-bound, this is especially important.

The other problem is the inefficiency in the DBB: there is no gain other than minimal disk space savings to using a lower-than-BLOCKMAX block size. Again, the DBB is sized for the largest block in the database, so records are being read in blocks into buffers which contain plenty of empty space.

So we have two problems to solve. The first one was caused by DBCONV, and the second one by DBSCHEMA. Because they both involve the same corrective action, we might as well attack them together:

1. Some master datasets exceed the other datasets by one or more sectors. They size the DBB so it must be larger, which wastes memory. They are inefficiently blocked, so they waste disk. They were made this way by DBCONV.

2. Some datasets (masters and details) have block sizes less than the others in the database. They are read into large DBB buffers with a lot of residual space, which wastes memory. They are efficiently blocked which optimizes disk, but their blocking factors are unnecessarily low, which impedes throughput. They were made this way be DBSCHEMA.

The solution to these problems is simple but may be very time-consuming. What needs to be done is to determine the best block size (the BLOCKMAX) for the database and set all the datasets to that block size. (You may not want to standardize your block sizes if you are very short of disk space and cannot absorb the resultant small increase in disk utilization.)

There are several important considerations in selecting a BLOCKMAX. Large block sizes permit larger blocking factors, which minimize disk I/O. The blocking factor is especially important for chained reads in detail datasets and serial reads in both masters and details. For example, a serial read of our sample master dataset would complete about 10% sooner with a blocking factor of 12 instead of 10. The blocking factor also has a significant effect on the amount of time it takes to do a DBPUT to a master set, and determines the impact of synonyms in master datasets because if a secondary (synonym) is located in the same block as its primary it can be addressed with no additional I/Os. This can have significant But a large BLOCKMAX yields a large DBB control block which may cause programs to execute more slowly because more memory is used for the DBB and its run-time buffers.

To correct the inefficiencies in a Turbo-converted database, we can size down the master datasets which were increased by DBCONV back to their previous sizes, thus reducing their blocking factors. This can be done with a DBUNLOAD/DBLOAD--when the database is rebuilt from the original schema, the block sizes and blocking factors of the impacted sets will be automatically reduced. For large databases, this can take hours or days. If you have a database restructuring tool, this task is infinitely quicker because the reblocking operation can be performed on just the affected master datasets rather than the entire database. This method will likely recover substantial disk space and memory space but compromize the blocking factors of the impacted master sets somewhat.

Another approach is to size up the other datasets in the

database to be the same as the block size attained by the increased master sets. This can also be done with a DBUNLOAD/DBLOAD, and will take about the same amount of time as the first method. If you have a restructuring tool, the time is reduced but far greater than the first method. This is because many more datasets require reblocking including detail datasets, which can take even longer to reblock because every pointer on every path must be recalculated and the related master chain heads must be updated. This method will likely utilize more space but no more memory, and will increase blocking factors throughout the database.

Or an entirely new BLOCKMAX could be chosen. Increased block sizes may be appropriate for systems with sufficient memory to support a large DBB control block and buffers. It is normally true that systems with an abundance of memory support many users, and those users generate a staggering amount of I/Os, and the disk drives are overworked and cannot keep up. These systems probably have ample memory to support large block sizes, which permit high blocking factors, and therefore less I/Os. The burden is therefore shifted somewhat from disk to memory.

In fact, although undocumented, the maximum block size was increased in TurboIMAGE to 4096 words (from 2048 words in IMAGE), allowing systems to make more use of the increased memory capabilities of the HP3000. This seems to be in theory, at least: Turbo DBSCHEMA does not actually permit one to specify a BLOCKMAX greater than the IMAGE limit of 2048--it considers this an error (we trust that this was an oversight by the labs. The default BLOCKMAX has not changed in Turbo DBSCHEMA: if none is specified in the schema by $CONTROL BLOCKMAX, a BLOCKMAX of 512 is assigned.

When choosing a maximum block size, beware of the increased memory requirements of the user control blocks (DBUs, called ULCBs in IMAGE). These can grow to up to 15 times larger than they were under IMAGE to support the multi-threading in TurboIMAGE. Because they are memory-resident, they can have a major impact on system performance; because one DBU exists for each user accessing a database, the memory requirements increase with the number of users. You should calculate the total memory requirement of the DBUs based on the expected maximum number of users--the formula is documented in Section 10 of the TurboIMAGE Reference Manual--and consider the decreased memory availability when determining the BLOCKMAX. It is more than likely that an additional megabyte of memory will be required to properly support TurboIMAGE, unless few users are accessing the database. (The DBU expansion, incidentally, is the single most resource-consuming change in TurboIMAGE which has the most substantial impact on system performance. It is not being discussed in detail here because, although it consumes a needlessly-

excessive amount of memory, there is nothing that one can do about.)

HP has documented the potential loss of efficient blocking in converted master datasets and some of the considerations in the TurboIMAGE Reference Manual and recommend a DBUNLOAD/DBLOAD to correct it. It also assures the user that "any potential problem of less effective blocking after conversion only affects (converted) IMAGE/3000 data bases. New data bases created in a TurboIMAGE environment will be blocked automatically by DBSCHEMA."

But wait a second! If DBSCHEMA assigned lower block sizes and blocking factors to conserve disk space under IMAGE, what does it do under TurboIMAGE? The TurboIMAGE Reference manual documents that "DBSCHEMA chooses a blocksize ... which makes the best use of disc space, and which may be substantially less than the maximum blocksize." So if this was not changed in Turbo DBSCHEMA, what will prevent it from doing the same thing when rebuilding the databases? Or when building new TurboIMAGE databases?

YOU must prevent it from doing so. This is something the TurboIMAGE manual does not document, so here it is: you can override DBSCHEMA's blocking by declaring your own blocking factors in your database schema. You can do this for all the datasets or only selected datasets by enclosing the blocking factor in parentheses on the dataset's "capacity" line as shown:

```
CAPACITY:    250000 (12);
```

This does not mean that you have to recalculate and declare blocking factors for every dataset in your database--you only need to override those which result in datasets blocked less than the BLOCKMAX. For an existing a database, do a :LISTF,2 to identify the lower-than-BLOCKMAX sets. For a new database, add a $CONTROL NOROOT at the top of the database schema which suppresses the creation of the root file but displays the table at the bottom which shows the "BLK LGTH": those which fall short of the BLOCKMAX by more than one media entry length ("MED REC") should be examined to determine whether the blocking factor could be increased--thus increasing the block size--while staying within the BLOCKMAX.

In doing your calculation, use the media entry length because it includes the pointers. You must also add in one or more words for the bit map which each block contains. The bit map size is determined by the blocking factor; a 1-word bit map is needed for blocking factors of 1-16, 2 words are needed for blocking factors 17-32, 3 words are needed for blocking factors 33-48, and an additional word is need

for each blocking factor increase of 16.

As a final note, remember that it is not necessary to do anything at all to your database after the conversion to TurboIMAGE--it will continue to function properly after DBCONV. But in order to optimize disk and memory utilization after conversion, while at the same time correcting long-standing under-blocking by DBSCHEMA, you should spend a little of your own time--and perhaps a lot of your HP3000's time--tuning things up.

Looking back at the trends we've seen in IMAGE, and now TurboIMAGE, although there have been improvements in the software in both functionality and throughput, it is definitely necessary to perform regular diagnostics and maintenance on your databases to keep them at their opimimum. Of course, a byproduct of these diagnostics is the identification and subsequent repair of breakdowns which can cause ill-affordable down-time and data loss.

Neither IMAGE nor TurboIMAGE are systems which can be left unattended--they run the life-blood of your company's data, and are too important to neglect.

THE NEW RPG3000

Gary Todoroff
Datamaster Computer Services
1735 'E' Street
Eureka, CA 95501
707/445-8425

ABSTRACT

As IBM expands the use of RPG through its System/3X computers, Hewlett-Packard has kept pace with its own RPG3000. While adapting to the IBM "standard", HP has also enhanced the language into a powerful applications development system, especially for on-line programs.

This paper first describes some of the history of RPG, comparing implementations of the most recent RPGII versions both by IBM and HP. Then the paper presents some of the new features of RPG3000 currently available on the HP3000 and also available on Spectrum software, Release 2. Technical topics include:

- SIGEDIT and RPG Screen Interface as an alternative to VPLUS
- Local Data Area (LDA) for program-to-program communication
- Automatic output positioning (relative end position)
- Full procedural files
- Function Key indicators
- EXCPT output by name
- File close and release options
- Data Structures
- $INCLUDE and RPGCOPY
- SORTA operation (internal array sorting)

PART I

RPG - SOMETHING OLD AND SOMETHING NEW

"RP-WHAT?"

In a recent interview I asked the just-graduated Computer Science major what he knew about RPG. "RP-what?" came the reply. Now why did he, along with so many others, know nothing about the language used in more mini-computer business installations than any other? Academia and mainframers seem particularly loathe to acknowledge RPG as a bona fide language. Even HP3000 folks tend to think of it as something of an antique. But RPG has come a long ways from its early Report Program Generator heritage to its present dominance of medium sized minicomputer applications.

RPG - AN OLD DOG WITH NEW TRICKS

Those who have heard of RPG may dispute its merits, but most will agree that RPG has a long history. Close on the heels of plug-board "programming", RPG started with the IBM 1401 and 360 of the early 60's, then IBM's System/3, the first popular mini-computer and father to the "S/3X" machines that would follow. RPG now runs on everything from small office environment S/36 computers to large S/38 database machines and a host of compatible computers, including the HP3000.

Computer forecasters are beginning to marvel at the longevity of RPG, but many HP3000 programmers don't know that it even arrived. However, RPG is alive and healthier than ever, not only in the IBM world, but especially within the HP3000 environment. While fourth generation languages are creating the headlines, RPG has been enjoying a quiet renaissance across many lines of computers, most notably the HP3000, including the new Spectrum.

HP KEEPING UP WITH THE IBM RPGII STANDARD

HP has been adding many improvements to RPG3000 lately. For example, it is HP's only language which uses an HP-supported screen interface other than VPLUS (discounting any ancient DEL/3000 screens). RPG3000 now adds a new approach with the Screen Interface Generator, SIG EDITOR/3000, which has been a part of HP's TRANSFORM/3000 conversion software for two years and is now a part of the RPG Utilities which come with the compiler. For conversions from IBM S/34/36 software, SIGEDIT provides an identical interface as the Screen Format Generator (SFGR) product from IBM.

Many other new features have been added to the language, mostly to keep up with the rapidly changing RPG standard established by IBM on the S/36. Many improvements are HP's own, displaying both originality and response to enhancement requests from the ever-growing community of RPG users on the HP3000. One of the more interesting cases involved the addition of a "TESTN" Calculation operation that tests whether the content of a field is numeric. IBM added the feature to RPGII **after** it was already a standard capability of RPG3000.

Bill Horst, Senior Development Engineer for RPG at HP, recently said that he no longer sees a trend for data processing shops to move away from RPG. "RPG has had an identity problem within HP for some time, but now I see a good future for RPG." Horst added that "The bridging of the PC gap is a key to RPG continuing as a current language," referring to the several new RPG compilers now available for the IBM PC and compatibles. HP's commitment to RPG is underscored by a flurry of product enhancements, a technical feat acknowledged in typical system programmer understatement: "It's a challenge keeping up with IBM," Horst admitted.

IBM'S BEST SYSTEM/34 WAS AN HP3000

Keeping up has kept three separate groups within HP busy for some time. Over three years ago, HP started the TRANSFORM project to develop a straight-forward approach for converting S/34 applications to the HP3000. A few third-party companies had already pioneered the conversion process, most notably McDonnell Enterprises of Issaquah, Washington, Info-Botique (now InfoCentre) in Quebec and Datamaster Computer Service in Eureka, California. At Datamaster, which was just me and a terminal in 1976, I converted applications from S/3 and the little S/32 to an early Series III. The conversion task became increasingly complex with the new screen interface (SFGR) introduced with the S/34. Even more difficult to convert was the IBM Operations Control Language. OCL is the "glue" that tied all of those RPG programs together with a programmer's toolbox of quick and easy ways to develop a friendly user interface. MPE's User Defined Commands (UDC's) paled in comparison with OCL, and available menu drivers didn't even come close. So instead of converting OCL at all, Datamaster developed ORBIX Control Language to basically run the OCL as it was written, while also adding a lot of new ways to use MPE.

Later, HP developed a similar OCL applications driver called PROCMON (Process Monitor), part of the TRANSFORM/3000 product. They also saw another incongruity between IBM's SFGR and HP's VPLUS, with no way to convert the completely different screen drivers used by RPG. So Craig Jester of Polaris in New Jersey was contracted to write an HP version of SFGR which became SIGEDIT. The outpouring of creativity surrounding RPG was even matched by the HP marketing department with a unique ad in magazines directed to the IBM market. The ad pictured a plain brown box marked "System/34 For Sale" next to a partially opened box with the HP3000 insignia peeking through.

THE MANY FACES OF RPG3000

As the TRANSFORM group developed an RPG compiler (Version 6.04) uniquely designed to run like an IBM S/34, the Language Lab in Cupertino continued with fixes and enhancements to Version 6.06, which HP customers receive when buying the standard RPG3000 compiler. The primary difference was no SIGEDIT screen interface in 6.06. Also, 6.04 lacked some of the new capabilities added to 6.06 by the factory compiler group. Rather than moving in parallel, it was more as if the two compiler groups were drilling from opposite sides of the same mountain, calculating to meet somewhere in the middle. Working on a new RPG3000 training course under contract to HP at that time, I did a lot of hiking over that mountain to peer into the still separate tunnels. Version 6.04 drilled deeper, acting more and more like a S/36, while 6.06 moved steadily inward, adding

many side-passages of new and unique features to ease the programmer's task. Finally, the two versions have met with RPG3000 7.0, to be released later this year, an implementation which should be very close to the capabilities of IBM's Release 3.0 of S/36 RPG. (IBM Release 4.0 added the new "indicatorless" Calculation programming - operations that allow Do-While, Do-Until, If-Then-Else and Case structured techniques formerly done with indicators — yet another challenge to keep up with IBM.)

The third and most recent compiler group involves the new Spectrum line of computers. The initial release of Spectrum will run RPG in compatibility mode. However, an entirely new RPG compiler is being written that will take advantage of Spectrum's 32 bit architecture and provide more flexibility for adding new features. As in all of the new Spectrum compilers, the technique of compiling source to a standard "U-code" implies compatibility across an eventual broad line of HP computers. Will we someday download object code compiled on an HP3000 for execution on an HP PC?

HP COMMITMENT TO TRAINING AND THE RPG STANDARD

All those new language features take training, and HP recognizes the need for teaching RPG programmers who bring a vastly different data processing background to the H3000 than the usual COBOL programmer. A brand new training course, "RPG3000 Applications Development", was just completed and is in alpha test. The one-week course teaches mainly by example, presenting a variety of DP problems, then showing specific solutions using RPG and MPE. Writing that course for HP gave me a chance to observe the Language Lab's special commitment to RPG and to maintaining the defacto standard set by IBM. That IBM standard is a fast-moving target lately, evidenced by the many recent announcements in the S/36/38 environments. IBM has been rapidly changing RPG as the gap narrows between the RPGII of S/36 and RPGIII on the S/38 with its database capabilities. The HP Language Lab is working hard to keep up with RPG's new features, while a lot of other computer companies are still in the dust of the outmoded RPG of the original IBM System/3.

RPG ON THE SYSTEM/3X "COMPATIBLES"

However, the arena of S/3X "compatibles" is growing rapidly, bringing RPG along as a tried and true language which keeps getting better with age. The word, "clones", previously applied mainly to PC systems. Now the word needs qualifying as S/3X clones debut. For example, the new Decision/36 from Decision Data Corporation runs both the S/36 and MS-DOS operating systems in one box with nary an IBM nameplate to be seen. "Baby/36" from California Software Products has also provided S/36 compatibility on PC systems, along with RPG compilers and S/3X style operating systems for the PC from Software West, Lattice, Inc. and Native Software. Wang and Prime promote source code compatibility with the S/34/36, and other mainframe manufacturers are jumping into the fray. The language common to them all is RPG.

APPLICATION SOFTWARE IN ABUNDANCE

Why such a surge in compatibility with RPG and the S/3X computers? The answer lies in the incredible amount of software which has been developed for S/3X. Over

4000 application packages have been written by software developers eager to tap into the huge market. The System/34 was reported to have achieved over 70,000 units in sales during its lifetime, and 39,000 are still running. On May 30, 1986, IBM announced the delivery of the 100,000th S/36. Earlier, in 1985, the S/36 PC announcement bridged the software gap between PC and mini, a revolution not yet as encompassing as the PC itself, but for RPG, even greater usage across a broad line of computers.

With such a huge base of installations running tested and mature software in RPG, what's a computer manufacturer to do? I believe the smart ones have already seen the evolving industry standard and have started to tap into that wealth of software developed for IBM S/3X. Today, software sells the hardware, and IBM S/3X software developers have the majority of the software. The only competitive solution is to make that software run on another piece of hardware, too. For example, an HP3000.

A FAST AND FORGIVING LANGUAGE ON THE HP3000

In many cases HP makes RPG programs run even better than IBM does. The efficiency of RPG on the HP3000 is a good example. Much of RPG source code does not compile directly to object code. Instead, tables are created from the source, which are passed to standardized object modules in the System Segmented Library (SL). The programmer who wrote these SL routines must have been very concerned with CPU cycles, judging by the execution speed of RPG programs. They run fast. RPG programs seem forgiving, too. Several years as a systems manager in a time-sharing environment revealed that inexperienced programmers could write equally sloppy programs in either RPG or COBOL. However, while bad COBOL would usually slow down the system, RPG seemed to often cover a multitude of sins. The RPG programs ran efficiently despite their coding faults.

RPG not only runs fast, it compiles fast, too. For an IBM to HP conversion project at Datamaster, we wrote a data entry and verification program (DENVER) that generates RPG source programs to emulate key-disk entry. One application comprising nine data entry record types and over 100 fields seemed sure to choke the compiler when DENVER created 5000 lines of RPG source code. (If you're not familiar with RPG, any program over 1000 lines long is considered quite large.) Instead, the compiler chewed on it about five minutes on a Series III, then spit out about 10 code segments, all adjusted neatly to 4k words. The program loaded in half a second and, even on a heavily loaded system, never slowed down the data entry clerk or anyone else.

CONNECTIVITY AND COMPATIBILITY

Of course, speed alone isn't reason to chose an HP over IBM, so why are businesses with RPG-based applications looking at the HP3000? Unlike the evolving IBM S/3X computers, the HP3000 was developed practically from the start to handle both office automation and distributed data processing. The machine had "connectivity" long before the term was popular. All the hooks are there to accommodate a huge variety of software, terminals, peripherals and other attached computers in a relatively easy fashion. On the other hand, the S/36 and S/38 use co-axial cabling with a synchronous protocol that requires expensive converters or emulation boards

to attach serial printers or PC's. Just to allow dial-up communication on a S/36 requires synchronous modems and an IBM interface in the $10,000 range.

Although upward growth compatibility is not quite the limitation it used to be on the S/36, the HP3000 still commands a much broader range of performance, especially at the high end. To even come close to a Series 70 or Spectrum 900 series, the S/36 DP site would need to convert to a S/38, which in many ways has less software compatibility with a S/36 than does the HP3000. Converting a S/36 to an IBM mainframe 4300 or 3033 is another world altogether.

A FEW LESSONS FROM IBM

However, HP can take a few lessons from IBM, especially regarding operating systems and user-interface. The System Support Program (SSP) operating system of the S/36 has been called bullet-proof, and "System Failure" is an unknown term in S/36 shops. SSP gracefully warns you of problems long before they become serious, while on-line explanations lead you if necessary to an error manual chock-full of explanations and potential recovery tactics. (An HP error manual like IBM's would cut HP's budget for PICS calls easily by half!) Even friendlier is the on-line "Help" capability of the S/36. Screens allow a fill-in-the-blanks approach to system utilities. Also, a help-key senses on which blank the cursor is resting, then displays a complete on-line description of that particular parameter. And an automatic file extension when a program hits end-of-file really saves some headaches. All together, the programmer benefits most, especially by not having to hide the operating system from the user, nor needing to anticipate a host of potential error conditions which the application program needs to trap. I hope as HP converts more S/36 sites, some of the good points will be applied to the HP3000.

LIMITATIONS ON THE SYSTEM/36

However a major weakness of the S/36 is the file directory. Although source and object programs are classified within "libraries" similar to a group/account structure, data file naming is extremely limited. Only eight characters are allowed to uniquely identify a file, this on a system that can now contain over a gigabyte of disc . A file can be subclassified with periods (e.g., GL.MASTR), but each period takes up one of the precious eight characters. Also the file backup commands are limited, with no equivalence to the STORE @.@.@ command. Data files, libraries, and data dictionaries (stored in "folders") must be backed up with separate commands. Also, no facility exists to back-up only the files that have been added or modified since a specified date. Amazingly, the file directory of the S/36 remains basically the same as the system on the original 13 megabyte-disk S/32.

CONCLUSION BEFORE WE GET TECHNICAL

Despite faults on both sides, the progress being made in S/3X compatible systems is remarkable. Without an ANSII standards committee or even a knowledge of one another as software development groups, competitive computer companies are developing RPG language standards that are both consistent and dynamic. I'm glad to see HP keeping up with the best of them and even more pleased to see their investment in a language which has been proven effective in such a wide variety of applications. There are times when you originate and other times when you just do a good job on an existing standard; HP has succeeded in doing both with RPG3000.

PART II

THE NEW TECHNICAL FEATURES OF RPG3000

HP has been very busy with RPG3000. Not satisfied to just keep up with the IBM defacto standard, HP has added some very powerful new features of their own. A tremendous number of complete business applications have been written in RPG for the IBM System/3X computers. It makes sense to be able to run that software on the competing HP3000 computers, so HP has invested a lot of effort to keep their compiler up-to-date.

This part of the presentation will briefly describe various new features of RPG3000. Then we will discuss the examples in detail. Without the benefit of chalkboard and handwaving, I encourage the reader to follow the examples along with a current HP RPG Reference Manual, preferably the January 1987 update (HP part number 32104-90001).

Here are some of those new capabilities which may be of use to the existing RPG shop and perhaps intrigue a few HP3000 users who have only heard about the language once described as: "RPG - A Fourth Generation Language Before Its Time."

FULL PROCEDURAL FILES

KSAM files can be processed both randomly by key and sequentially within key limits. Using both file access methods in the same program previously required the file to be defined once as chained and once as demand. The two logical files were then equated to the same physical file with a DSNAME file extension or with MPE file equations.

Now a Full Procedural File (F in column 16) allows you to define one file that can be accessed in Calculations both randomly (CHAIN) and sequentially within limits (SETLL, READ, READE).

```
FGLMAST  IF  F     256  8AI     2 DISC
              ^
              ^ F in column 16 allows file to be accessed
               both randomly (CHAIN) and sequentially (READ).
```

READE AND READP CALCULATION OPERATIONS

The READE and READP operations overcome a limitation of the READ operation. To read a file sequentially by key using a Demand file (D in column 16 of the File Specification) required two calculation operations: SETLL and READ as in the first example below. Also you needed to compare the key value of the record just read with the field that was used to do the SETLL (set lower limit).

READE allows you to read the demand file without doing a SETLL and compare. With READE the indicator in column 58 will turn on when the field in Factor-1 no longer matches the key value of the record just read. The indicator could therefore determine when the last duplicate key has been read in a KSAM file or when the end-of-chain has been reached in an IMAGE detail data set.

READP reads the previous record in a demand or full procedural file. In other words, it reads the file backwards. However, Factor-1 is not used, and the indicator in the Equal Subfield turns on when the beginning of the file is reached.

Old READ and SETLL method:

```
  FMASTFL  ID  F 256 256L15AI    25 DISC
  C            LNAME     SETLLMASTFL
  C            RLOOP     TAG
  C                      READ MASTFL                      20EOF
  C   20                 GOTO END
  C            LNAME     COMP LNKEY                       21
  C  N21                 GOTO END
  C*  ... DO OPERATIONS WHILE LAST-NAME EQUALS LAST-NAME-KEY
  C*                       .
  C*                       .
  C                      GOTO RLOOP
  C            END       TAG
```

New READE method:

```
  C            RLOOP     TAG
  C            LNAME     READEMASTERD                  20E-O-C
  C  N20
     .....(do conditioned output)
  C  N20                 GOTO RLOOP
```

DATA STRUCTURES

Now you can reformat fields and arrays without MOVE and MOVEL operations. Data Structures, defined by a DS in columns 19-20 of Input Specifications, allow fields to overlay each other. Modifying any field within the Data Structure simultaneously modifies any other fields that are defined within the same from and to positions of the Data Structure. Fields that overlap modify each other immediately within the program.

```
ITRANS    NS  01
I                                                .
I                                                .
I                                                .
I                                          71  75 KEY05
IKEY06        DS
I                                           1   1 ALPHA
I                                           2   6 KEY05

C* PUT "A" INTO DATA STRUCTURE WHEN PROGRAM STARTS
C N99                   MOVE "A"        ALPHA
C N99                   SETON                       99
C                          .
C                          .
C                          .
C             KEY06     CHAINFILE1                  60
```

LOCAL DATA AREA

One of IBM's best inventions on the System/3X is the Local Data Area. Like an ever-present mail box between processes, the LDA greatly simplifies program-to-program communication. On the HP3000, the LDA is a file automatically maintained by RPG programs. Just define the LDA and its fields within Input Specifications (no File Specification needed), and the program will read in the LDA at initialization. This happens before the first cycle, so fields are available at First Page (1P) output. When the program ends, any LDA fields that have been modified by the program are automatically updated and posted to LDAFILE.

To initialize the LDA file for your session, run RPGINIT in the PUB.SYS account. Typically a logon UDC can be assigned to run RPGINIT automatically at the beginning of a session.

```
ISYSCTL       DS
I                                           1   1 STATUS
I                                           2   9 HPUSER
I                                          10  15OMMDDYY
C...
C   03                  MOVE "X"        STATUS
C...
      .
```

RELATIVE END POSITIONS

RPG3000 now calculates end positions on output fields. Whether writing to disc or printer, just specify the fields without end positions in columns 40-44 of the Output Specifications . The compiler will automatically provide end positions according to the size of the field.

Spaces may be forced between output fields by using the RSPACE= parameter of the compiler $CONTROL record. This feature is especially handy for quickly programming simple reports.

```
$CONTROL RSPACE=3
FDISCIN  IP  F      80            DISC
FREPORT  O   F      80            LP
IDISCIN...
I                                          1  10 FLD01
I                                         11  20 FLD02
I                                         21  30 FLD03
OREPORT  H...
O                                        "FIELD 2    "
O                                        "FIELD 3    "
O                                        "FIELD 4    "
O        D...
O                          FLD02
O                          FLD03
O                          FLD01
O                                  +  2 "***"
```

RPGCOPY AND $INCLUDE

Parts of programs can be included in a program at compile time. The $INCLUDE compiler statement causes the file specified to be added to the program, especially useful for file Input Specifications or commonly used calculation subroutines.

```
$COPY
$CONTROL USLINIT
(File specifications, etc.)
$INCLUDE INSPECS1.SOURCE
(Remainder of program)
```

SORTA CALCULATION OPERATION

Program arrays are sorted with a simple SORTA operation. Just put the array name to be sorted in Factor-2 of the calculation specification and it will be sorted back into itself in either ascending or descending sequence.

SORTA cannot be used for numeric arrays. To sort in descending order, put a "D" in the Sequence Field (column 45) of the Extension Specification that defines the array. "A" or blank causes ascending order.

```
E                    ARY01       8  8
E                    ARY01A      8  8  A
IFILE01  NS  01
I                                          1  64 ARY01
C                     MOVE ARY01      ARY01A
C                     SORTAARSEQA
```

FUNCTION KEY INDICATORS AND SET OPERATION

Terminal function keys can be read by activating them with the SET Calculation Operation. In the first example, function key 8 is enabled with the SET operation. When the operator presses F8, the F8 indicator is turned on and the program will end.

In the second example, function keys F4, F7 and F8 are enabled with the SET operation. At the same time, by placing KEYLBL array in FACTOR-1, the function key labels are also downloaded to the HP terminal. Note that the array must be eight elements with 16 bytes in each element.

SET Example 1:

```
 C                       SET                         F8
 C                        .
 C                        .
 C                        .
 C   F8                  SETON                           LR
 C   LR                  GOTO END
 C                        .
 C                        .
 C                        .
 C            END        TAG
```

SET Example 2:

```
 E* FUNCTION KEY LABEL ARRAY
 E                     KEYLBL  1   8 16

 C                        .
 C                        .
 C                        .
 C            KEYLBL     SET                        F4F7F8

 O                        .
 O                        .
 O                        .
** KEYLBL ARRAY


  ADD    SRC CODE


  DEL    SRC CODE
  CHG     DESCR
```

FILE CLOSE AND RELEASE OPTIONS

Files can be dynamically closed from within an RPG program. The CLOSE operation in Calculation Specifications forces normal end-of-file actions to be performed earlier than the end of the program. Once a file is closed, the file cannot be reopened or reused for any further input/output operations by the program.

The Output Specification release option ("R" in column 16) operates in a special way for printer files. Releasing an output spoolfile will close it and reopen a new spoolfile to receive additional output.

CLOSE Operation example:

```
FSYSCTL  NS  01

C                     CLOSESYSCTL                H1
```

Release Output example:

```
FONEPAGE O    F      80              SLOWLP

OONEPAGE H...
O* LAST OUTPUT LINE TO ONEPAGE SETS RELEASE OPTION
O         DR  58   90
O                          TOTAL J   70
```

EXCPT BY NAME

The EXCPT Calculation Operation and Output conditioning can be done by literal name. Previously, exception output needed to be conditioned by an indicator. Now by specifying a name in the result field of an EXCPT operation, output is conditioned by the same name occurring in the Field Name of the Output Specification.

Also, a file name can be specified in Factor-2 along with the name in the Result Field. Note that in IBM RPGII the EXCPT name is specified in Factor-2 instead of the Result Field.

```
C                     EXCPT          ADDREC

C                     EXCPTSTDLIST   ERROR

OMASTER  E                ADDREC

OSTDLIST E                ERROR
```

SIGEDIT AND RPG SCREEN INTERFACE (RSI)

Fortunately for anyone who has ever converted IBM S/34 or S/36 programs to the HP3000, the RPG Screen Interface makes interactive programs run with hardly a change to the original RPGII. Unfortunately for this presentation, to discuss the features of SIGEDITOR and RSI in any depth would take at least another two or three articles.

However, a few comments are in order. I never have felt comfortable with the RPG interface to VPLUS. The buffers, calls to intrinsics and terminology are all too obvious as COBOL techniques stuck on top of RPG. VPLUS itself has many good features, especially the editing capabilities. But the hassle of harnessing VPLUS forms to RPG almost makes you want to go back to batch processing.

RSI has changed the RPG programmer's approach to interactive block mode programming. The programs read and write to forms in the same manner as to any other RPG defined file. Forms can be displayed at Detail or Exception output. Fields within forms can be formatted with Edit Codes or Edit Words just like output to a printer. Certain aspects of the output form can also be affected with indicators, such as changing a field to inverse video for highlighting an error. A line number can be passed from the program to the form, indicating on which line to begin displaying the form. The cursor can be positioned to a specific form field, too.

Many more features are available. See the SIGEDITOR/3000 section of the RPG Utilities Reference Manual (HP Part Number 32104-90006) for more details.

Here is a small sample of the RSI interface. The forms were created by SIGEDITOR in a file defined as DEMO5BFM in the File Extension Specification. The individual forms are called from the forms file at output time by specifying the name in the Constant field of the Output Specification. In this case, one form is used to clear a part of the screen (SCRN01). A second form displays data from a record just read from disc (SCRN02). A third form is appended to the form to display operator messages.

```
FWORKSIN UD  V     256            WORKSINR
F                                              KFORMS DEMO5BFM
E                    MSG     1   5 76                MESSAGES
IWORKSIN NS  01    1 CO    2 C1
I* DACCOUNT DATA SCREEN
I                                          1   2 RECID
I                                          3   8 ACINO
I                                          9  24 LNAME
I...
C                    EXCPT          R$CLRS
C                    EXCPT          R$DATA
C                    EXCPT          R$MESG
C...
O* CLEAR INFORMATION SCREEN
OWORKSIN E                 R$CLRS
O                                     K8 "SCN01    "
O* DISPLAY FILE DATA
O        E                 R$DATA
O                                     K8 "SCN02    "
O                          ACINO Z     8
O                          NAME       35
O...
O* DISPLAY MESSAGE SCREEN
O        E                 R$MESG
O                                     K8 "SCN99    "
O                N89       MSG,M#     76
```

A PARTING COMMENT

This will probably be my last article on RPG3000. Finally I'm giving up? No way! My recent work with Hewlett-Packard in Cupertino has involved their new product, "HP RPG". That's the new name for the RPG language due with the second software release for Spectrum. HP RPG will include the features just described here plus more that follow the evolving RPGII and RPGIII standards. Watch for more on the new HP RPG.

Evolving Performance Guidelines
- An '87 Update

Mark S. Tolbert
Customer Escalation Center
Hewlett-Packard CO.
2 Choke Cherry RD.
Rockville, MD. 20850

I. Introduction

The HP3000 world has changed dramatically since its introduction fifteen years ago, and today's CPU's and peripherals bear only a slight resemblance to the original members of the family. Just five years ago the largest HP3000 configurations were series 44's with four megabytes of memory, and sixteen disc drives (7925's or 7920's). Today a Micro 3000 with three to four Eagle drives can provide more CPU capability, more memory, and more disc capacity and throughput than these configurations. An even more vivid example is that fifteen years ago a series CX 3000 had a maximum of 64k bytes of memory, and this amount of memory required two card cages with 12 boards. The scaling of memory chips and their technology has changed so much that vendors are currently offering 16 megabytes of memory on a single board at prices not much more than the price of 64k bytes of memory 15 years ago. MPE has also gone through a number of major evolutionary steps in its lifetime. The common thread is an often emphasized, key strategy of the HP3000 world: programs written and compiled fifteen years ago can in almost all cases be restored on today's latest systems and MPE versions, and they will run. The "typical" customer application environment has also evolved and grown. There are many new subsystems, applications, and network environments. It should not be surprising then that many of the performance guidelines that were valid even a few years ago are obsolete today. The goals of this paper will be to discuss these changes in performance guidelines, and review the important issues to be concerned with today for improving performance on HP3000 systems.

The most obvious fact about the evolution of the HP3000 family and operating system is that perhaps its most important period of growth is beginning now with the emergence of Precision Architecture. The solution for many installations where there are performance problems, and where more room to grow is needed, lies in migrating to a Precision Architecture machine. Nonetheless, I believe it is also true that in many of these cases a higher performance system is not really needed, and the desired level of performance could be gained on the existing configuration by making relatively simple changes in operating strategies and application design. Following this tack, and working on squeezing out more performance on existing systems may not have the glamour of planning a migration to a series 930, but for many customers

it might be their best choice. The rest of this paper will deal with this "pedestrian" topic area. We will cover:

. basics on performance guidelines
. overview of changes in performance guidelines
. common problem areas today for system performance

The ideas presented here are the result of many discussions with other SE's and customers, and from experiences and training taken as an SE with Hewlett-Packard. It's not possible to credit everyone who contributed to the material presented here, but much is owed to many other people. The main point here is to try to provide a fairly comprehensive discussion of current performance guidelines for HP3000's; some points are fairly fundamental in nature - other points will probably be new and somewhat surprising to many HP3000 users.

II. Basics About Performance

Defining and characterizing the performance of systems is not a simple, straightforward matter. Some people talk as if computers were analogous to car engines, and performance was determined by the "engine" size and speed (word size? and MIPS rating?). However, the situation is more complex than this. Computer performance is governed generally by the interaction of three main resources: CPU capability, memory, and disc I/O capability. Processes can queue up waiting for any of these three resources, at which time a "bottleneck" will exist, and the overall system performance will be constrained. The goal is not to achieve near 100% CPU busy time, but to have these three main resources kept in balance with one another. By definition there will always be some bottleneck. Our goal is to configure the system properly, and match the system size to the number of simultaneous users and processes that will share the system.

MIPS (millions of instructions per second) only tells us something about the CPU's capability, if we assume the instructions on the different machines being compared accomplish the same amount of work. This is sometimes a very questionable assumption though, for operating systems can vary significantly in their efficiency, and one may require quite a few more instructions to execute the same amount of work as another. When we add to this discrepancies between the efficiencies of different file systems and database management systems, the CPU's MIPs rating shrinks in importance. Wordsize has even less of a direct relationship to performance. It affects the amount of memory that can be addressed in an instruction, and therefore segment sizes, but it has little to do with the speed of the processor. A 16-bit wordsize may have caused the writers of MPE to do some extra work in solving some problems, but for users designing their programs, the main irritant has been how to modularize large programs such that the stack size will be kept under 32k words. Performance is not usually the issue here.

In doing performance consulting, and delivering the HPSNAPSHOT product, we often find that the perception of system performance can vary substantially between different users, and the perception of the causes of problems can vary also. Records are not usually kept of response times and throughput, and what data is reported is usually of "impressions", not of objective measurements. HPTREND reports will help quantify a number of aspects of system usage, but it will not be possible to translate this data into data about "user transaction" throughput and response times. So it is not easy for a system manager to precisely characterize how his system is performing. Also, to keep a system performing in a manner that consistently exceeds users' expectations is probably an impossible goal; for one, standards will always get higher as systems perform better - we can always imagine things working a little differently and more efficiently. The definition of "good" response time will also vary between different applications and environments. Another key point is that it is the worst case, the peak times, that we have to be concerned about. A good average response time is meaningless if at peak times response times are unacceptable. If a system performs well 95% of the time, but unacceptably the remaining 5%, then it is very likely that the 5% interval will occur during critical periods where it cannot be tolerated.

III. Overview of Changes in Performance Guidelines

As we mentioned above, allowable memory sizes on HP3000's have grown dramatically, and this has changed some performance rules. When systems, on the average, had less than one megabyte of memory it was very important to watch segment sizes. The guidelines were that stack sizes should be kept under 12k words, and code segment sizes around 4k words. Today, these segment guidelines are rarely an important issue in solving performance problems. On most systems today there is usually some memory pressure, but this is due to disc caching demands; caching often uses all excess memory for its purposes. Algorithms used in MPE to prevent thrashing due to memory pressure had to be changed to accommodate the new effects of caching. The latest changes were implemented with the UA-MIT release.

It was also not very long ago that systems having performance problems were almost always suffering from a disc I/O bottleneck. With disc caching, this is rarely the case today. On most HP3000 systems today, using disc caching, the actual physical disc I/O rates will be well below the disc subsystem's potential. The first bottleneck that is reached as the job mix increases is usually a CPU bottleneck, or a memory bottleneck. A memory bottleneck is fairly easy to detect using performance tools, such as OPT, (or via the HPSNAPSHOT product), although the guidelines have changed somewhat with UA-MIT. In most cases, the problem can be easily solved by adding more memory, as the allowed maximums have increased generously. In fact, a case that needs to be explored is whether it is possible for a system to have too much memory: more extra memory than what caching really needs to be

effective. Although more performance testing needs to be done for this case, it seems possible that unnecessarily long caching linked lists could hurt performance.

So for many HP3000's today, the first bottleneck that will be reached will be the CPU. But what does this really mean? and what are the alternatives? First, if the CPU is busy nearly 100% of the time, that does not necessarily mean the CPU is overloaded. The CPU and discs will be busy 100% of the time (there will be no idle time) as long as there is at least one batch job always executing, regardless of how fast the CPU is. This can happen even though the on-line, interactive portion of the application mix is using a relatively small percent of the CPU. So the question to ask is, "is the interactive portion of the job mix using over 70% of the CPU?" If so, then the CPU is probably saturated just trying the handle the on-line users, and ways need to be examined to expand CPU capability, or reduce on-line CPU consumption.

If MPE is spending a large amount of time doing overhead operations, such as preempting and launching processes, then this also usually indicates the CPU is saturated. If this is more than 15% to 20% of total CPU time then it indicates there are probably too many processes contending for the CPU at once, and usually it will be found that a sizable number of "batch" type processes are part of the application mix. The reason batch processes present much more of a load for the CPU than interactive processes is that while interactive users usually spend over 95% of their time in "terminal read" state (comprised by time entering data on a screen, or viewing data on a screen), batch processes on the other hand have no "think" time, and are always ready to use the CPU. Even a "heads-down" data entry clerk, who can type 100 words per minute, will spend over 90% of the time in a terminal-read state. We will discuss this topic of batch processing specifically in section IV(D), but there are usually some solutions beyond just limiting the amount of batch processing. These solutions involve distributing the work to PC's or other processors, or handling the batch work more efficiently by using special tools, and with better access strategies.

If the CPU is overloaded just trying to handle the interactive mix, then our choices are to upgrade the CPU or reduce the amount of work the CPU must handle. We said above that on today's systems the traditional disc I/O bottleneck has been replaced by a CPU bottleneck, but to some extent maybe this is just a semantical point. The bottleneck is in many cases still handling disc I/O, but today we have largely put this load on the CPU through MPE disc caching, and taken it away from the disc subsystem. For high-end systems where the CPU is saturated, the logical step now is to remove the disc caching work from the CPU, and move this back to the disc subsystem using "controller" caching. This will put the three resources (CPU, memory, and discs) back into balance with one another. This entails replacing MPE disc caching with 7933 or 7935 controller caching - or the best option - 7937 controller caching. With the much higher density discs available today, and MPE disc caching, the trend at

most installations has been to reduce the actual number of disc drives and high-speed GICs. As high-end systems move towards controller caching, this trend should reverse itself, and more discs and I/O paths should be added to increase the potential amount of physical disc I/O that can be done. More attention will also have to be paid to spreading the disc I/O evenly between the available discs. In IMAGE, most I/O is done to detail sets, so try to identify the heavily accessed detail sets, and restore these specifically to different drives. (With Turbo-Image there is a "move" command in DBUTIL; ADAGER and other database utilities provide the capability to easily move datasets.)

IV. Common Problem Areas for Performance Today

In this section we will discuss a number of common mistakes and problem areas concerning performance. There are perhaps many systems with unused potential that could gain significantly better performance by correcting these mistakes. The solutions are usually fairly simple to implement. We will discuss the following:

- outdated default settings
- excessive logons and file opens
- improper configurations and disc space management
- "batch" processes
- malfunctioning devices, MPE problems, and application bugs

IV(A). Outdated Defaults:

- blocksize specifications
- buffer specifications
- caching specifications

The default settings used in the MPE file system and IMAGE for determining block sizes and buffer specifications were established when most HP3000's had less than one megabyte of memory, and far fewer users sharing the system. These defaults are usually not the best fit for most of today's configurations. Experimenting with the caching specifications can also frequently improve performance.

For Image and Turbo-Image, the default block size of 512 words, and the default blocking strategy, too often result in datasets having unnecessarily small blocking factors. This causes unnecessary disc I/O's to occur, and the unnecessary expense of some disc caching resources. Use the DBLOADNG program off of the SWAP tape, or the HOWMESSY program from ROBELLE, to determine what the blockfactor is for your key datasets. (Read the article, "How Messy Is My Database" in the "Image Handbook", for valuable information on interpreting the output of these programs - both programs provide the same output.) Do not undertake a massive job of reblocking all sets in your Turbo-Image databases, but identify those sets with small blocking factors (especially detail sets) that are accessed most heavily, and reblock these sets to a higher value

so that a higher blocking factor is achieved. In Turbo-Image, larger blocksizes in the range of 1024 to 1536 words, can often be used without unduly limiting the number of buffers that will be available. There used to be a problem that if the Image "Blockmax" size was increased much over 512 words, then when large numbers of users concurrently opened the same database, there would not be enough buffers available to support them efficiently. In Turbo-Image this problem is usually avoided because a separate "extra data segment" is set aside just for the buffers. Using tools such as DBCHANGE, ADAGER, DBGENERAL, etc. can save unloading and loading the entire database to accomplish this reblocking operation, and some utilities, such as ADAGER, also provide help in choosing the right block size.

Default Image and Turbo-Image buffer specifications are also inappropriate for most databases today. That is because today databases are often shared by a varying number of users, and the defaults cause the extra-data segment for the buffers (the DBB) to be frequently rebuilt as this number of users fluctuates. When this occurs the database is made inaccessible to all users. Use DBUTIL to change the buffspecs to a constant value for a wide range of users. For example, execute the following command in DBUTIL:

```
SET XXXXXX BUFFSPECS = 25(1/120).
```

This would set the number of buffers to 25 (which is usually adequate for even the most heavily accessed databases) for all ranges of users, and would eliminate a significant amount of unnecessary overhead from occurring. In pre-Turbo databases, a smaller value may have to be used to allow enough room in the DBCB for the lock area, and to prevent an Image status error of 62.

Letting MPE choose the blocking factor for sequential files that are used regularly in applications, is usually not a good idea either. Your applications may use flat files more than is apparent (temporary work files), and in many cases these files have blocking factors that are very small. This may cause unnecessary physical disc I/O's to occur. (A block factor of "one" will cause a physical I/O to occur for every logical record that is read.) Disc caching can lessen the harmful effects of these mistakes, but we can improve performance even more by blocking the file properly in the first place. If the record size is not excessively large, we can often get away with using a block factor of 128 or a multiple of 128. This ensures the blocks finish on sector boundaries without wasting any space.

For MPE disc caching, performance can often be improved by increasing the "random fetch quantum" to a value such as 45 sectors. The problem is that the random fetch quantum is used by all read operations in the Image, Turbo-Image, KSAM, and Sort subsystems, even when the files or datasets are being accessed serially. These subsystems were written long before caching arrived on the scene, and they only use the "FREADDIR"

intrinsic. Experiment with the random fetch quantum, and use the SHOWCACHE command to see whether more disc I/O's are eliminated by choosing a larger value. This may especially help for nighttime batch processing (96 sectors may be a good fit here). Another benefit is that this will decrease the total number of cache domains MPE needs to manage which can improve performance. People often ask if there are any cases where disabling caching for writes (CACHECONTROL BLOCKONWRITE = YES) can improve performance. Experiments we have done indicate that it does not. Even though disc caching is not nearly as effective for write operations as it is for reads, it still provides some benefit. We do not save any caching logic from being executed by disabling caching for writes. The "blockonwrite" option exists purely for attempting to ensure more file integrity in the case of a system failure. The only case in which it could affect performance is where memory is limited, and by disabling caching for writes the number of cache domains would be decreased; memory pressure would be reduced.

Making these changes in blocksizes, buffer specifications, and caching specifications reduces the work that must be done by the CPU, and also in the disc I/O subsystem. In doing this the demand for memory will be increased. In today's world this is a very favorable, economical trade.

IV(B). Excessive Logons and File Opens

When performance for a system begins to degrade, it is often times the logon procedure and application start-up process that cause the most frustration. Once users have applications running, their response times may be much better. This is understandable given how much overhead is associated with logging on and opening files. At logon time, numerous system resources must be exclusively accessed and locked, system tables must be updated, and a large number of disc I/O's are executed to set up the process. File opens alone often entail a significant amount of disc I/O. (Opening an existing permanent disc file may entail doing as few as three disc I/O's, but opening a new file with 20 extents may require over 25 disc I/O's to be executed. Most of these will be updates to the disc free space map.) Therefore, for a system that is apparently overloaded, one of the least desirable things users can do is frequently logon and logoff, or constantly switch between applications. Applications should be set up so that users do not have to logon to different accounts to run their principal applications, and users should be encouraged to stay in the particular application they are executing as long as possible, and get as much work done as possible, before switching to a different application. A typical logical transaction in a database application may use around 30 msecs. of CPU time and do 10 physical disc I/O's. On a system that is not saturated, a number of users can simultaneously execute this type of transaction and obtain response times under 5 seconds. In contrast, let's look at the case of logging on to a new account with system, account, and user UDC's in effect, and with a logon UDC that starts up an application that opens 10

files. This may require several seconds of CPU time alone, and hundreds of disc I/O's to be completed. Also, we may have to wait to acquire access to the loader (only one process can use this at a time), and to acquire locks to other key system resources such as the directory and file system SIR's (System Internal Resources).

An application design strategy that can eliminate most of this "start-up" overhead is to use process handling techniques to have an application front-end process open up a number of different terminals and present a menu on these terminals. Instead of logging on, a user would pick one of the menu options, and the "father" or "control" process would "activate" the specific "son" process on the terminal being used. A significant amount of work would be required to design and program this type of strategy, but in certain cases it is a good fit. (The HP "MM", Materials Management, application uses this type of strategy.)

As we implied above, UDC's can be part of the performance problems involved with logging on. Avoid the situations where UDC's exist at the system, account, and user levels. Also avoid cases where multiple UDC files are used at these levels. Combine the UDC's as much as possible into common files, and be stingy with what UDC's are allowed. Make sure COMMAND.PUB.SYS has a blocking factor of 128 or 256 and that the individual UDC files have high blocking factors of around 200. (For further details read the article in the May 1985 INTERACT magazine written by Roberta Estes on UDC's and performance.)

Avoid using UDC's in batch jobs: especially avoid using system-wide or account-wide UDC's in batch jobs (or by any low-priority process) when the UDC's are executed from a file shared by high-priority interactive processes. In these cases, it is possible the low-priority process will either directly or indirectly impede many processes on the system as it tries to execute the UDC's. As the low-priority process executes a UDC, it must lock the file-control block of the shared UDC file, but if the system is busy chances are good the low-priority process will be preempted before it has finished its work with the UDC file. It will then impede any other process trying to access this file, but unfortunately, these processes may also have other key resources locked, and the chain of impeded processes will grow quickly. The situation will not be remedied until the low-priority process finally gets scheduled to execute again, and completes its work with the file-control block. This may be a significant period of time on a busy system.

There is another point to consider in setting up UDC's that eliminates considerable overhead and maintenance. You can avoid creating individual UDC commands for all the heavily used subsystems in PUB.SYS by creating one UDC which will work for all of them. Consider using the following UDC:

```
R PROG
OPTION LIST
RUN !PROG.PUB.SYS
****************
```

(We could run QUERY.PUB.SYS by typing: R QUERY.)

It is also fairly common for user application programs to have "performance bugs" built into them. The most common variety of this problem is for a program to do unnecessary file opens. In one performance study I worked on last year, a program updating a large database opened and closed a temporary file each time within the main processing loop. Unbeknownst to the programmer this caused his program to do over 320,000 physical disc I/O's to the free space map in a four hour period, while the program only did 100,000 disc I/O's to different datasets. Similar mistakes have been noted where programs open and close files (such as a formsfile) each time within the main transaction loop. Within Transact, for instance, be careful with subroutines "calling" other subroutines. The calling subroutine's files (those defined in its "System" statement) will be automatically closed during the call operation. Therefore, if the main transaction loop encompasses this type of subroutine call, there will be excessive file opens and closes. Of course, it is not easy for a system manager to monitor how frequently different files on his system are being opened, but one possible method to collect data is to activate logging for "file close" events, and report this activity using LISTLOG5. Files showing unusually high file open and close rates may indicate there is a problem with one of the applications accessing them. Tools that performance specialists use to monitor systems also capture these statistics.

IV(C). Improper Configurations and Disc Space Management

It should not surprise anyone that an improper I/O configuration, or system table configuration can cause performance problems. It is surprising though how many installations do have mistakes in these configurations. Configurations need to not only obey MPE guidelines and the hardware guidelines, but should also be set up with performance in mind. Try to implement these guidelines:

- The top priority is to spread the discs evenly across the high-speed GICS that are available on your system. (An IMB can have two high-speed GICS). For example, do not put all the discs on one GIC, and the other high speed devices, such as a laser printer and 7978's, on another. Secondly, distribute the other high-speed devices evenly across these GICS. Lastly, low-speed devices such as INP's can be put on these GICS, but that is usually not necessary. GICS are inexpensive, and the low-speed devices can be distributed across their own "low-speed" GICS. (The GICS aren't any slower, they are given this designation depending on what devices are connected to them.)

- If low-speed devices share a GIC with discs, then be sure the low-speed devices have a higher DRT number than the discs. Also, the CE should ensure the "slot" priority is lower for the low-speed devices (only for 3000/4x and 3000/5x systems). These are fairly minor points, but are easy to implement.

- For 3000/68's and 3000/70's, have at least two IMB's and at least three to four high-speed GIC's. Something is probably mismatched if you have a 3000/70 that needs only one IMB, and where there are only two or three discs (the disc I/O structure of a fairly low-end system).

- do not mix CS80 discs (793x, 791x) where RPS in enabled, with MAC drives (792x) on the same GIC.

- on 3000/4x systems and 3000/5x systems, use ATP's in place of ADCC's as much as possible to eliminate overhead associated with character interrupts.

- to help performance, use pairs of 2334's connected to ATP's in place of MTS to 2333's, and for remote high speed printing. This can eliminate a significant amount of CPU overhead required by MTS software (MPMON).

Fragmented disc space on your system will also impair performance. Run FREE5.PUB.SYS at least once a week to check the free space. If it shows there are not any large free areas available, (100,000 sectors or more on a 793x drive), then the discs need to be recompacted. Free areas of 100 sectors and less are, for the most part, wasted spaces, since few file extents will fit in this size area. The disc free space can be reorganized by either doing a RELOAD (do an ACCOUNTS reload, followed by a complete RESTORE off of a ZERO dump tape, first turning on caching), or by doing selective VINIT CONDENSE operations on separate discs. The time to do these operations needs to be scheduled carefully; a complete RELOAD often takes over 5 hours depending on the number of files and disc space used on the system, and a VINIT CONDENSE operation usually takes about 45 minutes on a 793x drive. While the VINIT CONDENSE operation is under way the system will be locked up and inaccessible to all other users. The point to emphasize though, is that if the system manager neglects doing one of these operations to reorganize disc space on a regular basis (say once a quarter), the discs can get fragmented, and this will cause the file system and MPE to do a lot of unnecessary extra work. Extents of files will be located far apart from one another causing longer disc seek times. Performance can degrade very noticeably.

A number of years ago HP Systems Engineers often recommended that the system disc, LDEV 1, be put in its own device class (usually SYSDISC), so that it would be kept free of user files. This was once a good idea, because otherwise LDEV 1 would have to solely take care of all the MPE I/O to the directory and virtual memory, and also it would have to take care of its share of user I/O too. Today with disc caching, much of the directory I/O will be eliminated, and virtual memory can be spread to all discs so it is not necessary to treat LDEV 1 as a special case. If LDEV 1 is left in a device class of its own, it will probably be underutilized.

System tables should be monitored also; if a table reaches 85% utilization or more the table should be increased at the next opportunity to prevent the possible case of running out of table entries. If this occurs performance on the system may degrade, because processes usually will be impeded as they wait to obtain entries in the table. In some cases, a system failure will occur (SF602 due to no SWAPTABLE entries being available). With UA-MIT, and the use of AUTOALLOCATE, the LST, CST, CSTBLK, AND CSTX tables may continually show near 100% utilization, which is the way it is supposed to work with AUTOALLOCATE. You can use TUNER5 or OPT to monitor system table utilization.

IV(D). Batch Processing

There is confusion about how batch jobs should be used, and even what the definition of "batch" really is. We will define "batch" to mean those processes which require very little, or no user "think" time. Since batch processes require minimal user input as they run, they are very disc intensive and CPU intensive. With disc caching they are especially CPU intensive. It follows from this that batch processes do not fit in well with multi-programming concepts; they do not share the system well. Little can be gained by trying to run multiple batch jobs simultaneously. We will pay extra CPU costs by having MPE manage their attempt to run concurrently, and manage their contention for common resources. In almost all cases, the aggregate time for running a set of batch processes will be less if we run them sequentially, one after the other, rather then all at the same time. Although politics don't always allow it, the most efficient way to handle batch processes in the disc cached world is to keep the job limit very low. Some batch jobs voluntarily pause for periods of time (such as SLEEPER), so these do not present a problem.

As we implied above, batch processes are not just those processes that are executed in the "DS" queue, or started using the STREAM command. Many installations have "covert" batch processes running in the "CS" queue, and these start at the priority of the interactive processes. Some common examples of CS queue batch processes are:

. compiling large source files interactively
. compiling a large VPLUS formsfile
. executing long database queries
. transferring large files over a datacomm line
. doing raster file conversions of graphics files
. executing the TDP formatter for large documents (via a "Final" command)

These processes will quickly be penalized by MPE for not doing a terminal read within the "average short transaction time", but they can still have an adverse affect on the "real" interactive users (those users expecting response times under 5 seconds). Also, remember that even the batch jobs that are executing in the "DS" queue (or "D" queue) may indirectly have a negative effect on the interactive users, because high priority MPE routines will be executed on their behalf to service their interrupts, and perform other overhead work necessary for them. For instance a batch job doing a transfer of a large file across a DS line, will indirectly use the services of DSMON which runs at the priority of 30. A sizable amount of DS traffic could easily use over 10% of the CPU's resources at this high priority.

There are usually good solutions to handling all the batch work that must be done on systems today. For database batch processes, using products such as SUPRTOOL, and ASK can significantly reduce serial access times against large datasets. In one case I worked on, a customer was able to reduce an 11 hour batch report job that needed to run overnight, to 3 hours using SUPRTOOL. Using "autodefer" in Turbo-Image, or "output deferred" with Image can also reduce access times significantly, especially when the datasets are blocked properly. For intensive office functions, and graphics operations, the solution is to download this work to PC's. Similarly, for batch work that must be done against a database that interactive users need constant access to, a possible solution is to distribute the database to a second system using SILHOUETTE, and do the batch work on the second system.

IV(E). Malfunctioning Devices, MPE Problems, Application Bugs

Sometimes systems mysteriously seem to start performing very slowly, but the problem is not due to the system load, or the inability of the system to keep up with the job mix. There are some types of hardware problems, MPE problems, and even application specific problems which can cause these symptoms. Fortunately, these problems only occur very rarely, but it is important to know about them for they can look like a severe performance problem. For instance, there are cases where a modem, a terminal, or other device will begin sending a stream of control characters that cause interrupts, and the CPU may become saturated just handling this stream of interrupts. The system will appear to be nearly locked up, and will run very sluggishly. (If you have OPT, you can probably detect this condition by examining the global menu, and seeing if the CPU time in the overhead category is substantial - probably over

50%.) Another example is that there are known problems with auto-dial asynchronous modems, when they are not owned, causing MPE to cycle through the steps of opening the port and the TTIO file, and speed sensing the port, because the modem has DSR and CD strapped high. Excessive overhead can be used doing this for multiple autodial modem ports, and an ICS stack overflow could also occur (causing a system halt). Subtypes 10 and 11 were developed to handle this problem for ATP ports. A fix has also been incorporated into UB-Delta 3 to put the termtype files (such as TTIO) into an extra data segment; since they are no longer files, they will not be repetitively opened as users log on.

There are potentially a number of different types of problems in this general category, and it can sometimes take involved trouble-shooting to isolate the problem. In a perverse way, it usually helps that in most cases the system performs so slowly that it appears hung, and a memory dump is taken. Although memory dumps only catch the last instants of what was happening in the system, and important pieces of data might have already come and gone, they usually do help solve these problems. (This especially true for U-MIT and later versions, where virtual memory is also captured.)

The following is a list of some of these types of problems.

I. Hardware
   - . interrupting devices
   - . excessive modem handshakes
   - . slow GIC

II. MPE
   - . Any number of possible problems can surface with a new MIT. As enhancements and fixes are incorporated into new releases, bugs can also be created which impair performance in certain cases. There will be patches and workarounds available for these problems, but the problems can vary from release to release. It also can happen that a new release may bring bring out a user application bug that didn't cause any problem, performance or otherwise, in the previous release of MPE.

III. Applications
   - . improper locking strategies
     - - improper use of MR (causes process deadlock)
     - - improper database locking (usually locking at high-level around terminal read)
   - . user process in B-queue takes CPU
   - . improper process-handling design (father and son processes lock up most of system as they are caught in loop activating one another)
   - . Image master sets fill to over 85% of capacity

V. Conclusions

We have discussed how HP3000's have evolved, and how these changes have caused the typical performance bottlenecks to change. In the last section we have also talked about some of the common problem areas for performance on systems today, and made recommendations to solve these problems. The underlying theme is that on many HP3000 systems today there are ways the system performance can be improved. Often times this entails making very few changes to the hardware configuration. System managers can identify and solve many of these problems themselves, but investing in performance consulting can often times provide a payback that far exceeds the cost of the consulting. The HPSNAPSHOT product can be used by a performance consultant to identify where the system bottleneck is, and provides detailed information about all the processes active on the system when the collection is taken. For each process it can be determined: how much CPU time the process used, how many logical and physical disc I/O's were done and to what files, how many transactions the process executed, how much of the time the process was impeded, and what were the principal reasons causing the process to stop (i.e. was the the process preempted frequently or blocked for I/O, etc.). With custom performance consulting, specific applications can be analyzed to determine if they contain any "performance bugs". Along with the HPSNAPSHOT tools, programs such as SAMPLER, may be used to collect data on these application programs. It may be discovered that unnecessary file opens and closes are being done, that an improper locking strategy is being used, or that unnecessary stack expansions and contractions are occurring due to improper calls to ZSIZE. The HPCAPLAN consulting product can be used to model the effect on performance of different workloads that are being anticipated, and the effect of different upgrade configurations for these workloads. Problems can be anticipated well before there is a crisis. For some performance problems it requires the use of sophisticated tools and a specialist's experience to solve the problem, and the above performance consulting products will be well worth the effort and cost.

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.
- Provide 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:

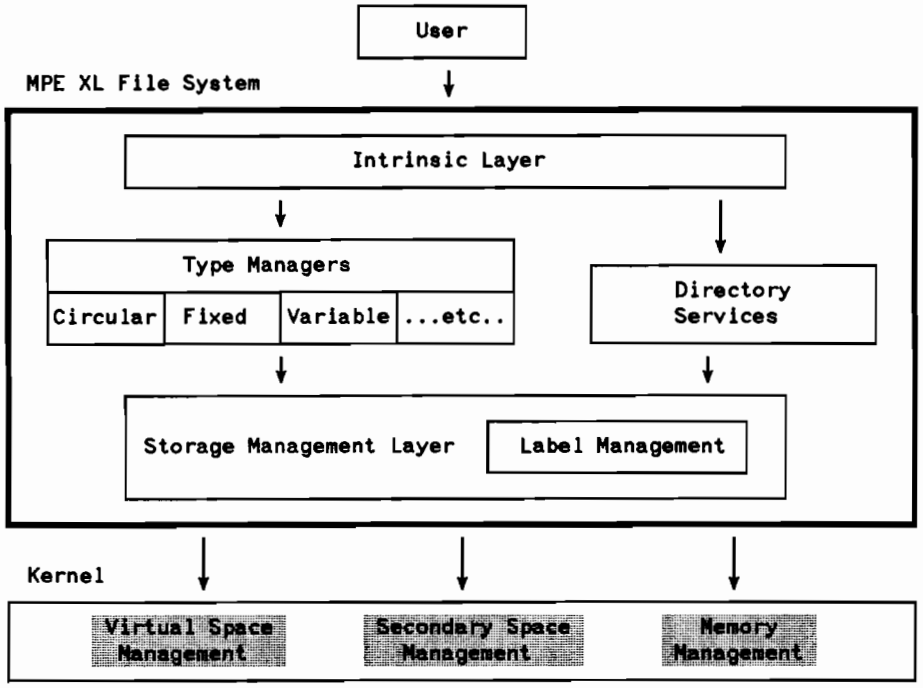
  1) Automatic space and file recovery after a soft crash.
  2) Optional hard crash recovery in case of disc head crashes.
  3) Common services to database and users through transaction intrinsics.
  4) Dynamic file backup with users concurrently accessing files.

- 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 direct access to mapped I/O files, a method which provides for 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. The externals of XM will 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 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, or 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 bracket 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 a deadlock and resolve it by backing out one of the transactions and 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.

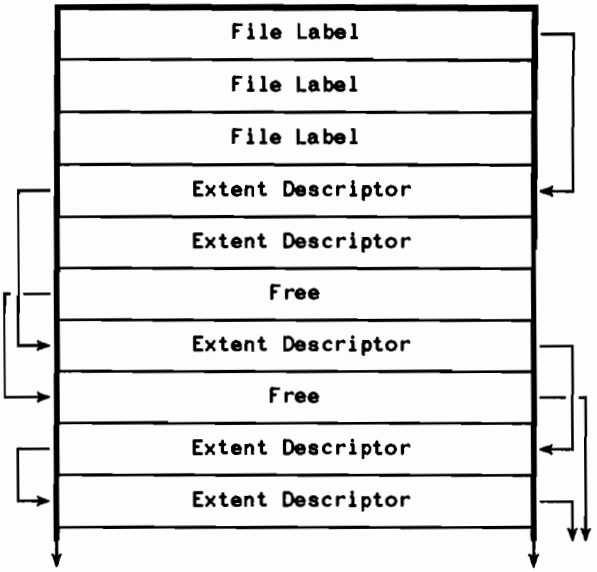
It's not enough to provide just these data integrity enhancements. In a heavy disc transaction environment, *concurrency* must also be maintained to effectively utilize the processing power available. Multiple users on potentially multiple SPUs must be able to gain access to the data at the same time. 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.

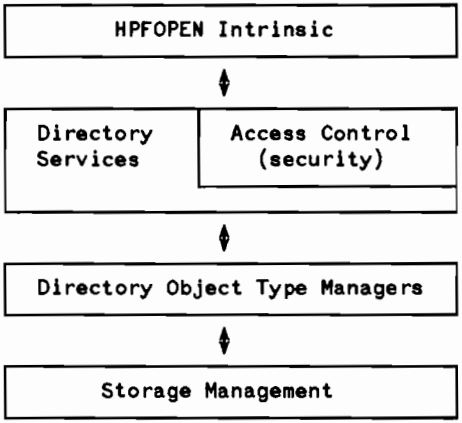
There are many benefits to be had by maintaining file labels and extent descriptors in this way. The Label Table can be very large, limited essentially by available disc space. It follows then that a single file can have practically an unlimited number of extent descriptors and therefore extents. In 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).

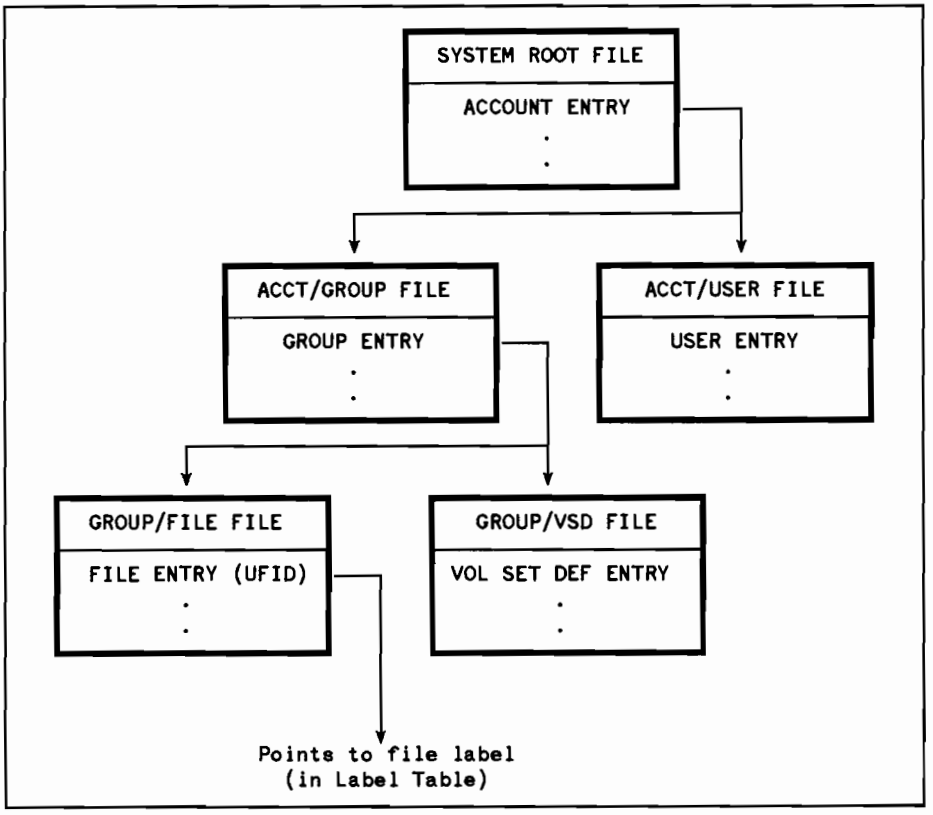
While maintaining complete compatibility with current MPE, the new services and data structures provide for many extensions and eliminate a number of past artificial limitations and hindrances on system availability:

- 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 names beginning with a common string.
- 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.
- Directory objects (nodes) are spread across discs, resulting in better I/O parallelism and system robustness (creating the 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 MPEXL_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.

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.
- Ease of use and minimal operator intervention, especially for "office environment" configurations.
- Backup of data distributed within a network.
- Backward and forward file transport capability between MPE and MPE XL systems.
- Backup of the system directory on STORE tapes.
- Supported 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)

In the first release of MPE XL, the DYNAMIC, LOGONLY, and STORESET options will not be available. On later releases, 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.

HEWLETT PACKARD

Capacity Planning With Standard Workloads

Knowing Tomorrow's Performance Today

Interex '87

Tim Twietmeyer
Performance Technology Center
Hewlett-Packard Company
8010 Foothills Blvd., Roseville, Ca 95678

Introduction

One of the most common questions presented to HP performance specialists is, "Can I add this application to my system, and if so, how many terminals can I support before I need an upgrade?". In the past, this question was difficult to answer because there was no tool provided to predict the impact of a new application or an increase in terminals to an existing application. With the advent of KASANDRA, an analytic modeling tool, and Standard Workloads, it's possible to model the impact of a new application on any HP 3000 system.

This paper will discuss the process used to create Standard Workloads and how they are used with the KASANDRA analytic modeling tool to predict response time, throughput, and server utilization. These tools are used by HP performance specialists to deliver the HPSNAPSHOT and HPCAPLAN consulting products.

Workload Composition

The first step in the capacity planning process is measuring the amount of work the system completes during a fixed period of time. The term *workload* refers to the amount of service demands imposed on a system by a certain set of programs, commands, and data while it's completing its required tasks. These service demands draw from the finite amount of system resources that are available to complete the work, primarily CPU and disc.

On the 3000, this measuring of service demand is completed by using several tools that monitor resource utilization. These tools are responsible for measuring the amount and type of I/O that is being completed, the time the CPU is busy, the number of transactions completed, and sampling other areas of the operating system and hardware that impact system performance. These tools include some HP products and some internally developed tools to aid in the workload measuring process.

For most HPSNAPSHOT and HPCAPLAN measurements, the duration of the measurement is one hour. The objective is to capture the workload when the intensity is very high and representative of the system that's to be analyzed for capacity planning. One hour works well because it is long enough to capture the phenomena required for the study, yet short enough to keep the amount of captured data to a minimum. Collections shorter than an hour don't collect enough data to

accurately represent the workload and collections longer than an hour tend to include periods of low system usage.

When the measurement is complete, the system workload is broken into smaller identifiable components. This decomposition of the system workload is required because modeling throughput and response for a system-wide workload has little significance for systems which run diverse applications. The workload is broken into *classes* of work representing the different types of work which were completed during the measurement interval. These goal of this step is to identify the different classes such that the classes represent work that had comparable service demands, require a separate class for modeling, or need to be distinguished by various organizational units (e.g. accounts payable, manufacturing).

This definition of these classes is accomplished by referencing a workload definition file. Each class defined is the workload definition file is identified by a unique name, is assigned a category, language, and a set of program files that will be bundled to compose the class. An example of a class definition is:

```
CLASS=TEXT AND DATA PROCESSOR/3000
CATEGORY=OFFICE AUTOMATION
LANGUAGE=PASCAL
TDP.PUB.SYS
SCRIBE.PUB.SYS
TDPSP@.PUB.SYS
```

Each class is also differentiated by whether is was run as a job, session, or as part of the operating system, and also by the queue in which it executed. If a program file is not defined in any class definitions, the file name is used as the class name. Of these self defining classes, only those that consume over ten percent of the CPU or disc space are isolated in their own class. The remaining self defining classes (<10%) are bundled into a single class defined as *OTHER.

When the breakdown of a system workload is complete, the workload might be represented by the following classes::

Class	Type	Queue	
FCOPY/3000	J	C	
FCOPY/3000	S	C	J - Job
HPDESK	J	D	S - Session
HPDESK	S	C	O - Operating System
HPWORD	S	C	
*OTHER	S	C	L - Linear
SYSTEM	O	L	C - C queue
SYSTEM	S	C	D - D queue
SPOOLER	O	L	E - E queue
TDP	S	C	
AP.PROG.FIN	S	C	
PAY.PROG.FIN	S	C	

For each class of work, the workload definition software must also allocate to each class the appropriate amount of resources that they consumed during the measurement. This is done by examining the log files which were created by the measurement tools and assigning the proper

amount of CPU, disc I/O, memory requirements, and any other performance related resources to the class which consumed them. Ideally, every CPU second and I/O should be traced to the process that initiated it.

At the completion of this class definition and resource allocation, the information is saved for comparison with other workloads. This requirement spawned the creation of the Summary Data Base (SUMDB) where all workloads are saved for each HP site collected. This data base has one data set named WORKLOAD that stores the workload definition and the assigned metrics. The elements included in the data set are: CPU instructions, physical and logical I/O by type (MPE, Image, KSAM, Directory, and Memory Manager), think time, number of terminals active, the percentage of disc I/O's to each disc LDEV, and other workload related metrics.

Analytic Modeling

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 are explained in *Quantitative System Performance, Computer System Analysis Using Queueing Network Models, Edward D. Lazowska, John Zahorhan, G. Scott Graham, Kenneth C. Sevcik, 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.

The workload information collected and refined during the workload composition phase can now be used as input to KASANDRA and projections can be made for server utilization, throughput, and response time. The workload data stored in the SUMDB is extracted and the appropriate values are assigned to the required modeling parameters. These parameters define characteristics of the workload and must fit the structure of the model.

An example of the model that Kasandra represents is shown below:

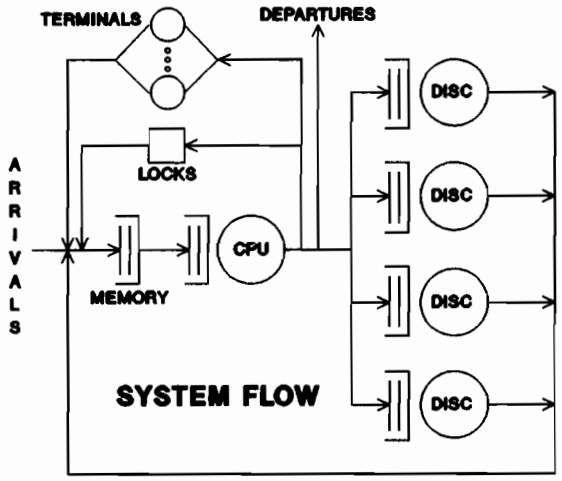


Figure 1. KASANDRA Model Flow

The figure shows the different customers or transactions and how their demands are made on the various service centers. A transaction arrives in the model and must queue for memory and the CPU. After the CPU services the request, the transaction could depart from the model, be serviced by one or more of the four disc drives, or return to complete a terminal interaction. For each service center, the workload definition must provide a description of the demand on that center.

The workload description can be input manually into KASANDRA or passed through an MPE file. Special workload organization software has been written to organize the workload and create the appropriate KASANDRA commands. This particular software is not discussed here, but it allows a user to interactively regroup and organize classes and pass this information to the modeling tool.

An example of a workload description for the model is shown 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                          Create JOBS
CPU=          582.25,70            CPU=          818.788,70
Disc=       28769.609              Disc=       40515.886
Drive=  1,     41.90               Drive=  1,     33.33
Drive=  2,     33.35               Drive=  2,     33.33
Drive=  3,     24.75               Drive=  3,     33.34
MPL=            0.407              MPL=            5.496
PRI=1                              PRI=3
Type=Transaction                   Type=Transaction
Thruput=       0.000278            Thruput=       0.000278

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 classes. Each class is defined by the CREATE command followed by the class 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[2] defines the degree of software constraints for the class. 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 class. 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 classes defined and predict the response time, utilization of devices, and throughput. These values are compared to those output in a report by the workload organization software, 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 class, or perhaps disc service times. If the values calculated by the model are not in the acceptable range the modeler must review the workload definition and attributes to identify why the results are so diverse.

An example of a KASANDRA model output is shown below:

Used .254 Processor Seconds

Class	Response	Throughput	CPU Util
JOBS	7343.5	1.0	41.2
USER	.9	1207.2	11.6
SYSTEM	.7	315.9	2.1
OFFICE TEXT PROC.	1.0	203.1	1.6
GRAPHICS	1.2	16.5	.1
PROGRAM DEVELOPMNT	1.3	279.3	1.8
NON HP UTILITY	2.0	28.7	.7
HP UTILITIES	2.1	134.2	2.7
DATA COLLECTION	1.5	17.2	.2
OS	235.4	1.0	1.1
		Total CPU	63.0

The measured CPU utilization for this particular model was 62%.

KASANDRA provides several forms of output. A report is output showing utilization, response time, and throughput for each class. 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.

[2]Simply defined, MPL is the number of active threads of control provided by a class. 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.

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.

Standard Workloads

Often customers would like to know how a particular software product will perform even before it is implemented. If we store hundreds of collections of HP system in the SUMDB, why not use this data to predict the performance of a system that isn't currently running a certain HP product. For example, a customer may want to know if their current system can handle the addition of twenty HPDESK users without adding additional CPU power. The customer has yet to purchase the product, but they would like an estimate of the impact of a typical HPDESK workload.

By analyzing the information stored in the WORKLOAD data set of the SUMDB, the modeling parameters required by KASANDRA can be generated. The WORKLOAD definition file groups each of the required program files into HP products. Each product has its own entry in the WORKLOAD data set, and by looking at each entry as a product and not as part of a workload, statistical analysis can yield the parameters that are required to represent a typical product workload.

The data flow of the standard workload process is shown below:

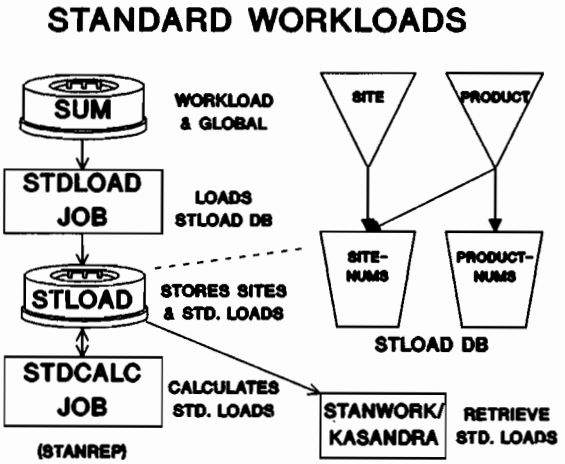


Figure 2. Standard Workload Flow

A standard workload data base is created (STLOAD) to store the specific information pertinent to generating the KASANDRA class parameters. The job STDLOAD extracts data from the WORKLOAD and GLOBAL data sets of the SUMDB and places them in the STLOAD data base. For each entry added to the SITE-NUMS set, a summary record for that product is flagged in PRODUCT-NUMS indicating that a new entry is available to be included in the calculation of a

standard. The STDCALC job reads the PRODUCT-NUMS set looking for entries that require summarization. When a summary is required, each entry is read from the SITE-NUMS data set and a sequence of statistical calculations are executed. The statistical results are placed back in the appropriate PRODUCT-NUMS entry. This summary record can then be displayed from within KASANDRA and the product parameters added to the model being evaluated.

Two different methods have been applied to analyze the workload data. The first method treats each product collection as one sample and the intensity of the workload is not weighted. In this case, a product collection that completed five thousand transactions has the same impact in the calculations as a product collection that completed five transactions. Each site's CPU, disc, and think per transaction values are read and the mean, median, high, low, and standard deviation are calculated. Once the mean is determined, each site's values are read again to determine if any site has values outside three standard deviations from the mean. If so, the site is eliminated and the calculations are executed a second time. This calculation is completed for each product by job type and queue in which it executed, and a minimum of six product collections is required for a product to be considered for a standard calculation.

The second method treats each completed transaction as a sample. This causes the product collections that completed large numbers of transactions to have a larger impact on the standard calculations. With this method, all transactions and their corresponding CPU, disc, and think values are totaled and the mean is calculated by dividing these totals by the sum of the transactions. There are no outlying entries eliminated and a minimum of six product collections is required to establish a standard workload.

The results of the two methods on a sample product are show below:

	Method 1	Method 2
CPU Low	0.00	0.00
Median	0.09	0.09
Avg.	0.10	0.10
High	0.57	4.70
Disc Low	0.00	0.00
Median	4.30	4.52
Avg.	5.48	6.06
High	28.30	263.44
Think Low	0.40	0.40
Median	11.40	11.62
Avg.	11.60	10.10
High	24.80	26.51
Terminals	4.98	4.84
Avg. Trans.	1719.5	1664.1
Modeled Trans.	1507.8	1674.4
Collections	73	76

For this example, method two models closer to what we expect for the throughput of an average HP 3000 site. Other examples tend to show similar results for terminal workloads. Other filtering techniques such as eliminating product collections that have a low number of completed transactions, or only including collections from similarly configured systems will be investigated.

With this type of information, a performance specialist can estimate the impact of a new product on a system that doesn't presently use the product. By using the CPU, disc, and think values and applying the proper number of interactive terminals, a user of the modeling tool can investigate the response, throughput, and CPU utilization that can be expected with the product addition.

The concept of standard workloads is not an attempt to find the PERFECT workload to represent an HP product. It is intended to be used as a reference to see how customers are applying our software products on their systems. It is clear from the collected data that each customer uses the product differently and each user interacts with the product in their own unique way. With this in mind, each application of this standard workload concept must consider the customer's application of the product, the experience of the user population, and the size of the system for which the product is intended.

Additional results of this study will be presented when the paper is delivered at the conference in September.

HEWLETT-PACKARD'S COMPANY-WIDE NETWORK SOLUT'

CYNTHIA URROZ
HEWLETT-PACKARD
CUPERTINO, CALIFORNIA

INTRODUCTION

Hewlett-Packard's company-wide network solution is based on the X.25 industry standa., which has been adopted by all major computer and telecommunications vendors. The X.25 standard ensures multivendor connectivity now and in the future, as well as lasting value for current hardware and software.

The network provides a complete range of services for on-line interactive access to remote information and electronic mail. It also offers efficient data transfer between distributed processors, mainframes, minicomputers and personal computers regardless of where they are located.

This paper examines the three components of the company-wide networking solution: 1) X.25 access products, which allow computer systems, workstations and terminals to access remote applications over an X.25 network; 2) SNA access, to allow HP-to-IBM and IBM-to-IBM communications over the network; and 3) the backbone network -- switching equipment that provides a multivendor X.25 transport network. Although this paper does not cover them, Hewlett-Packard also offers BSC access products.

Providing the right information to the right people at the right time can improve your company's competitive position by cutting your operational costs, boosting productivity, and increasing customer satisfaction.

The typical company-wide network environment reflects geographical dispersion, equipment from multiple vendors, rising datacomm costs and integrated applications. Within such an environment, HP customers expect multivendor connectivity, network control and reliability, cost control, and security.

g those expectations is the goal of HP's company-wide network solution. Currently are seeing an emphasis on dedicated networks that incorporate multiple proprietary rchitectures with point-to-point connectivity and access to public data networks (PDN). HP is positioned not only to meet current user needs but also move into the 1990s by providing multivendor connectivity on a single network, using a backbone based on the X.25 standard.

WHY X.25?

X.25 offers a reliable protocol based on industry standards, full routing capability, and optimal use of transmission link capacity. These features translate into data integrity in a multivendor setting, high availability, and cost effectiveness. HP's strategy for addressing company-wide networking needs is thus based on X.25 with integrated network management. This is the foundation on which we've built our Private Packet Network (PPN).

The X.25 standard offers other features that make it a strong foundation for wide area networking. It enables users to optimize cost/performance considerations since it enables them to share links and use both public and private networks in whatever combination meets their needs. In addition, Private X.25 networks allow the owner to have complete control over the network, including managing its growth.

ACCESS PRODUCTS

For companies with widely dispersed sites and a need for data exchange capabilities, Hewlett-Packard offers access solutions that enable users to interconnect with remote systems across the backbone network, across a public data network, or a combination of the two.

For medium to large networks with both batch and interactive traffic requirements, HP offers X.25 system connections on HP 3000s and HP 1000 computers. This enables terminal

users and software processes to log on to remote systems, transfer data, and access remote files, databases and peripherals.

For small networks with well defined needs, HP's point-to-point system connection is accomplished by the NS Point-to-Point Link. The Async Serial Network Link is best suited for access to multiple systems over telephone lines as a low-volume, low-cost, low-speed alternative.

For remote user connections to a central host computer, HP offers a wide range of alternatives - async connections over dial-up lines for limited communications, or leased lines for heavier traffic loads. With AdvanceLink, a remote PC can transfer files to an HP 3000 or access a full range of HP 3000 applications. This solution doesn't require any extra hardware investment and is easy to install. In addition, public PAD connections are available for dispersed users or worldwide communications.

For terminal clusters whose users generate high traffic and need high reliability, HP's 2334A Plus can function as a statistical multiplexer or an X.25 cluster controller, both over an X.25 network. This network access product is designed for users with several co-located terminals, high connectivity needs, character and block mode applications, and a need for access to public and/or private X.25 networks.

HP-to-IBM COMMUNICATION

It's common for a company with geographically dispersed sites to need communications with an IBM mainframe at corporate headquarters. Updating a database, accessing an IBM 3270 application, and electronic mail are common requirements in this environment.

Through SNA access to IBM mainframes over both SNA/SDLC point-to-point lines, and public or private X.25 networks, Hewlett-Packard provides access for HP 3000s.

A variety of networking configurations is available for HP-to-IBM communications. For SNA over an X.25 network, HP provides SNA gateway software that runs on a headquarters-based HP 3000, converting X.25 to SNA when accessing IBM mainframe applications. Another alternative entails SNA/X.25 protocol conversion via a synchronous PAD connected to the X.25 backbone; this converts SNA to X.25 and back again at the remote end.

Other host-based SNA products include SNA emulation software, such as SNA NRJE, SNA IMF, LU 6.2 Base, and SNA Link. The same kinds of SNA emulation that are done through the SNA Gateway are also available as standalone products. HP 3000 users who want access to IBM mainframes, where there are no plans to use public or private X.25 networks, can do so over an SNA network.

BACKBONE NETWORK

HP's PPN is made up of a family of packet switching nodes, Network Control Processors, optional Distributed Control Processors and Network Operator Consoles.

The PPN switching nodes form the backbone of the X.25 network. They provide the X.25 access points into the network and handle all traffic routing. HP's family of switching nodes is based on a modular design offering a wide range of connections, from 8 to more than 500 ports. The HP PPN provides connectivity to all the company's equipment, through X.25, asynchronous, SDLC and BSC interfaces.

In addition to the different interfaces, the availability of a public X.25 gateway and high port modularity mean excellent connectivity. Features such as the PPN's multi-processor architecture, end-to-end routing, call congestion control, class of traffic routing and distributed control processors add up to outstanding performance.

Reliability is provided by on-line automatic sparing and dynamic routing, as well as the HP PPN's optional redundant network control system. Call establishment verification and network activity logging mean high network security.

Network control and maintenance are also important considerations, and the HP PPN provides these through several important features. Among these are: on-line configuration, automatic statistics collection, both local and remote disagnostics, "hot" module replacement, software downloading, and distributed operator consoles.

PROTOCOL ANALYZERS

In a multivendor environment where X.25 is the common denominator, HP's family of protocol analyzers and test equipment provides important monitoring and support capabilities. Each analyzer is designed for a different environment, with different features and characteristics. But all five have common operating, setup, remote transfer and display characteristics. The protocol analyzers use a softkey-driven menu, human interface to provide sophisticated testing capability. Troubleshooting, system integration and optimization of network performance by the analyzers add an important dimension to HP's datacomm support. HP's X.25 network performance analyzer enables data center managers to spot signs of network degradation before users are aware of trouble.

NETWORK SUPPORT

For the last six years, Hewlett-Packard has been a fixture as the top-ranked vendor for customer support in the annual Datapro survey. Very little that the company could say about support speaks as strongly as this six-year endorsement from users.

But the company isn't content to rest on these laurels, especially since the growth of networking and network technologies gives a new meaning to support. HP offers a diverse set of support services:

Network Planning and Design puts the networking expertise of our Network Consultants to work to analyze network requirements and develop detailed network design.

Network Prepare offers recommendations on a plan for implementation, covering scheduling, staffing and operational procedures to ensure a timely installation.

Network Startup provides coordination assistance for installation, testing and documentation to get the network up and running quickly.

NetAssure is the name HP gives to its post-installation support. Once a network is operational, HP offers troubleshooting and management of problem resolution in a multivendor environment.

SUMMARY

Hewlett-Packard's company-wide networking solution addresses the issues involved in making information available to everyone who needs it, whether the people are across the office or on different continents. Based on the international X.25 standard, HP's networking solution is designed to meet multivendor needs now and in the future.

EFFECTIVE KNOWLEDGE BASE DESIGN

R. E. VAN VALKENBURGH
AMPEX CORPORATION
P.O. BOX 190
MARVYN PKWY.
OPELIKA, AL 36803-0190

The current trend of computer hardware economizing will only succeed in placing further pressure on those creating the software for the computer systems.

One of the areas with continuing potential for increased software development productivity is in effective data design. There is empirical evidence that classical 3rd generation programming productivity can be increased many-fold when good data structures are used. Many 4th generation languages are severely limited by (and possibly unusable with) poor data structures. Good automated system design will never achieve its full fruition without well designed data structures.

This paper treats the logical design of structures, and provides a technique for data normalization which is perhaps the simplest technique published to date. The technique is one which lends itself well to computer aided software engineering (CASE) and automatic code generation.

CONTENTS

The KNOWLEDGE BASE

Similarity with Data Base

The similarity between the term "Knowledge Base" and "Data Base" is no coincidence. The Knowledge Base concept that will be unfolding here incorporates many of the qualities we have become accustomed to in the usual concept of the data base. After some consideration I concluded that the term "Data Base" is too confining to describe the practical but powerful method of software design and implementation treated herein.

Charles Lewis described the difference between the concept of "database" and a "data base" as follows [Lewis]:

Database

> Database is (1) a systematic methodology for the standardization and integration of data resources at an organizational level, or (2) all of the data or the organization which is organized or controlled using a database methodology.

Data Base

> A computerized data base is a set of computerized files on which an organization's activities are based and upon which high reliance is placed for availability and accuracy.

I would like to "hitchhike" upon those concepts and paraphrase Lewis by defining the concepts of "Knowledgebase" and a "Knowledge base":

Knowledge-base

Knowledgebase is (1) a systematic methodology for the standardization and integration of information and knowledge at a particular level of organization, or (2) all of the knowledge and information of an organizational level which is organized or controlled using a Knowledgebase methodology.

Knowledge Base

A Knowledge Base, on the other hand, is the complete set of computerized representations upon which an organizational level's activities are based, and upon which a high reliance is placed for availability, accuracy, and completeness. A Knowledge Base is the logical and physical representation upon which the Knowledgebase methodology is implemented.

More than a Data Base

A Knowledge Base incorporates, as a subset of itself, a full data base implementation. But in addition to the structure and data contained in the data base portion, the Knowledge Base also contains in its entirety the constraints upon that data and all of the relevant facts about that data.

Fully Integrated

In simple terms, a Knowledge Base contains within it a data base and all of the logic necessary to inquire, add, modify, and maintain the information within it. A possible implementation could perhaps be a fully integrated data dictionary, data base, and set of all of the related code to process the data.

Knowledge Base Examples

Today we have a number of examples of software packages which might qualify as partial implementations of the Knowledge Base concept, such as some of the various spreadsheet programs and the programming language ProLog. Unfortunately these implementations, besides being incomplete Knowledge Base implementations, are not yet suitable as solutions to entire commercial systems.

Solution Today

With sufficient attention to the methodologies, this Knowledge Base concept can actually be implemented today as a solution to existing commercial business system requirements with a suitable data dictionary, data base management system, and well designed module oriented code. The disadvantage here is that high dependency is placed upon the implementors to avoid constructs which are unnecessarily complex and redundant. The ultimate Knowledge Base would not permit that possibility.

Software Gains

The purpose of the Knowledgebase methodology and the underlying Knowledge Base, is to begin to make significant progress against the persistent software problems that have plagued the industry since the advent of computers, and to form a solid foundation in support of the continuation of the information explosion.

The SOFTWARE PROBLEM

Hardware Progress

Since the 1950's great progress has been made in computer systems. The gains made in hardware improvements and economizing have had an almost incomprehensible impact. It is these gains and economies which have led to what many term the "information explosion", and has led to the increasing pervasiveness of computer systems and an ever larger reliance upon them.

Software Progress

Unfortunately, over the same thirty year period, our advances in computer software has been far less impressive. The consensus is that software costs are excessive and continue to be plagued by serious quality and reliability problems. Software tends to be just too complex.

In the words of James Martin [Martin A]:

> With non-trivial programs, however, there are ... too many paths for complete testing. Furthermore, changes to the program, which often have to be made, cause unpredictable effects. COMPLEX PROGRAMS REMAIN A MINEFIELD IN WHICH WE CAN NEVER BE SURE THAT ALL THE MINES HAVE BEEN REMOVED.

Software Costs to Computerize

The costs of developing and maintaining software is thought to be at least 70% of the total costs of computerization. And if anything, the proportion of software costs to the total computerization costs are expected to rise further as subsequent economies in hardware are realized, and the labor intensity of software development and maintenance continues to swell.

Significant Effort

In large part today's software requires significant maintenance effort due to the poor maintainability, quality, and reliability of the software.

SOFTWARE COST BREAKDOWN

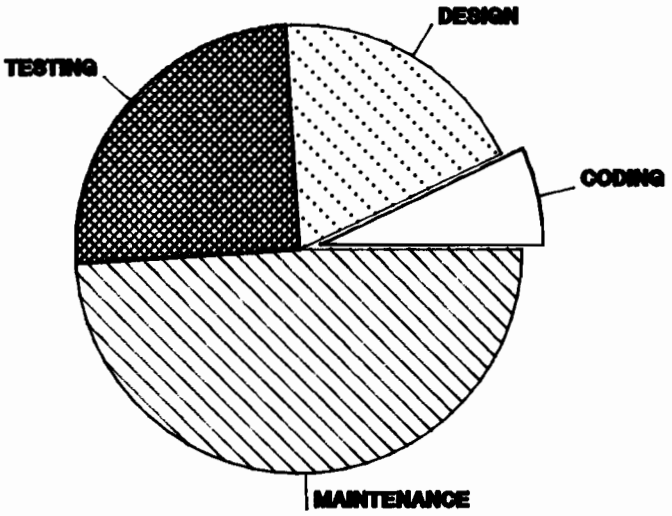


fig. 1

TODAY'S SOFTWARE COSTS

Software Industry

Over the years software development and maintenance has grown into a very large and expensive industry involving greater and greater numbers of people. The actual annual costs of software are not precisely known, as detailed statistics are not available, and many companies desire to keep their disappointments to themselves. But in 1979, the annual expenditures on software were estimated to exceed $20 billion [Boehm A].

Industry Expenses

Why are the expenses of computer systems is difficult to obtain? According to Daniel Melcher:

> The rules of the computer game are that you only talk about what you are going to do, never how it turned out. This is a science in which you publish the results of your experiments before you make them. I have tried to get and to publish stories of the sad aftermath of many noble experiments, but the trouble is that the victims won't talk. [Melcher]

Industrial Tragedy

Robert Heller points out one of the few tragedies that were expressed:

> Univac even sent some hardy pioneers to Europe to publicize its achievement at a plant in Marietta, Georgia. The Georgia customer, too, added some of its men for the ride in its pride over this ultimate in management control systems. The wonder was designed to reap extra profit from tight control by, among other things, giving a running account of actual expenses and updated estimates of production costs, keeping the program on schedule and even building in automatically the financial results of any change in specification.
>
> The Georgia company was Lockheed Aircraft, and the project the C5A monster transport, overran its budget by $2 billion, worked none too well at the end of the process, and would have sunk the company without trace in 1970 but for a Washington rescue act. [Heller]

It is not hard to understand why there is no desire to present this type of problem for public scrutiny.

U.S. Govt Software Costs

Much of the information we have related to the cost of software has come from the U.S. Government, most notably the Department of Defense and the National Aeronautics and Space Administration. The U.S. Government is the single largest computer user and software developer. To an uninformed person, what we learn about the costs of software from the U.S. Government is simply a state that the rest of the industry is unwilling to verbalize.

Testing

The U.S. Air Force spent some $750 million on software testing alone in 1972 [Naur].

Cost Per Line of Code

In 1976 the software cost of development per line of code was estimated at $8, for code of average complexity [Lecht]. But the costs of more complex code were reported to be about $75 per line of code in development, and an astronomical $4000 per line of code in maintenance [De Roze].

Maintenance Costs

The software cost of maintenance (the fixing of errors and adding changes after installation) and the consequential testing can be expected to exceed 75% of the cost of the actual software development. The SAGE military defense system, for example, with an initial development cost of $250 million, continued to cost an average of $10 million per year even after ten years of operation [Boehm B].

Continuing Problems

After 12 years of operation, the U.S. Strategic Air Command's 456L Command System continued to average one software error per day [Thayer].

Apollo Program

Probably one of the most adequately funded and properly planned software projects was for NASA's Apollo lunar flights. It was able to lure some of the best talent. Testing methods were as thorough as possible (much more so than industry customarily has the luxury of performing). Competing teams were set up to ensure adequacy of testing and development. In total over $660 million was spent on the development and testing of the software, but almost every significant fault in the Apollo program was directly the consequence of software errors [Ulsamer].

The NATURE of SOFTWARE PROBLEMS

Translation Errors

If we wish to categorize the major cause of software errors, there is no doubt but that most software errors are attributable to errors in translation.

Elaborate Process

The production of software is an elaborate translation process, beginning with a specification of a problem and ending with a set of detailed instructions that achieve a computer solution to the problem.

The job of software specialists (whether they be analysts, designers, or programmers) is one of translating the original problem or its subordinate representations into various other subordinate representations, until a computer executable result is produced.

Error Prone

This whole method is one that is prone to many potential errors in translation and interpretation.

Communi-cation Problems

Translation errors begin immediately in the specification phase of software development. In general, users have substantially different backgrounds than the data processing personnel with whom they interface. When one individual communicates with another of common backgrounds, it is a simple, unconscious, and almost automatic process to "make up" for gaps and mistakes in the communication process; it is an easy matter to make assumptions, to "read between the lines".

Inter-personal Impediments

Interestingly enough, during communications between people of notably different backgrounds or frames of reference, the human capability to "read between the lines" can be a serious impediment to effective communications. This inherent ability of the human animal often induces a failure in recognizing that specifications are incomplete, ambiguous, inconsistent, or laden with errors.

As James Martin so eloquently stated [Martin B]:

> Human beings can invent, conceptualize, demand improvements, and create visions. They can write music, start wars, build cities, create art, fall in love, go to the moon, and colonize the solar system, but they cannot write COBOL or Ada code which is guaranteed correct. They need automated tools that will translate their desires into computer code. And because most of their desires are not computable, the tools must help in successively refining their requests until they are computable.
>
> ... [The tools] need to have the property that they translate broad human thinking about requirements into a computable form and successively refine it until it is possible to automatically allocate the resources needed and generate bug-free code.

Much effort has been expended on improving the process of software development, but in general the results have been a disappointment. Repeatedly the necessary changes have been resisted, but perhaps more often the emphasis on improvement has been placed in the wrong areas.

RESISTING CHANGE

When will we ever learn?

Fortran Language

In 1956 the Fortran programming language was developed as the first "third generation" programming language of many third generation languages yet to come. Fortran continues to be widely used, and although improvements have been made over the years to the language, it is substantially the same as earlier versions. There are still more individuals who "know" the Fortran programming language than any other computer language.

Mariner I Loss

But what does Fortran do for us more recently? The now famous loss failure of the Mariner I probe to Venus was said to be due to a single programming error in a Fortran program. The statement

```
DO 3 I = 1.3
```

was interpreted by the Fortran compiler as a simple assignment statement, assigning the undeclared variable "DO3I" the value of 1.3. Since the Fortran programming language does not require the explicit declaration of variables and ignores embedded blanks, the programmer's error (typing a period, instead of a comma) went undetected, the variable "I" was never initialized, the loop never executed as intended, the Mariner was forever lost in space, and the rest is history.

As president Reagan launches the Strategic Defense Initiative (more commonly know as "Star Wars") I hope it has occurred to someone not to tolerate programming with a language that permits such obvious (but obscure) errors!

Deficiency Known

Actually, to our current dismay, this particular deficiency of the Fortran programming language has been well known since long before the Mariner loss. Many other third generation programming languages have been designed to eliminate this type of error (by using reserved words and requiring all variables to be predeclared, for example). In fact this deficiency was probably well known and understood by the time COBOL was released in 1960.

Bad Policy

But now, thirty years later, we continue to standardize practices that we know to be "dangerous" and error prone.

WRONG EMPHASIS

Productivity Tools

The latest buzzwords have been "productivity tools" and "fourth generation languages". For a long time the jury has been out on the final consequence of theses "improvements". While there is no doubt that they can have an notable impact on software development costs, they are not as likely to have such a constructive impact on the largest component of the software cost (maintenance and testing).

Fourth Generation Languages

On fourth-generation languages, Donald Campbell, Director of Marketing and Product Planning, Cincom Systems, Inc. points out:

> ...Hundreds of software tools have resulted from advances in computer and software technology. Vendors claim that these products will be compatible with existing systems and increase performance, productivity, and user satisfaction. Very few of these tools, however, truly fulfill this promise. [Campbell A].
>
> Although productivity improvements are possible using almost all fourth-generation languages, the results can be misleading... One of the prime productivity factors of a fourth-generation language depends on reducing future maintenance costs by decreasing design errors... faulty design or specifications can be blamed for 83 percent of maintenance time. Only 7 percent of all maintenance can be traced to coding errors." [Campbell B].

Efforts to Increase Productivity may Increase Costs	In many cases, the efforts to increase programmer productivity may actually increase the software development costs because they have only succeeded in making programmers design and code faster, and have not succeeded in developing and enforcing methods which will cause software to be designed with more precision and accuracy.

The SYMPTOMS of SOFTWARE PROBLEMS

The symptoms of software problems are many, but let me enumerate some of the more prevalent:

* System requirements are often in a constant state of change

* Specifications do not adequately embody the requirements, and are usually incomplete, ambiguous, inconsistent, and incorrect

* Users do not fully comprehend the specifications and therefore cannot verify the accuracy and completeness

* Analysts and programmers do not fully comprehend the requirements and therefore cannot verify specifications for accuracy and completeness

* Conventional programming is too labor intensive and error prone

* Program testing takes too long, and errors often continue long after testing is complete

* Programs do not fulfill the specifications and therefore do not fulfill the requirements of the end user

* Program maintenance tends to obsolete the specifications, and to continually reduce the quality of the code

* Programs contain excessive redundancy

* Simple changes to existing programs often cause floods of unanticipated problems

* The same data is redundantly retained and is represented in incompatible forms

* The design and implementation tools that do exist are far from integrated

* Purchased software and systems software are of poor quality, incompatible with other software, and big resource sinks

The SOLUTION to SOFTWARE PROBLEMS

Solution Proposed

Many solutions have been proposed and attempted to alleviate the software problem. I propose another here, embodied in the fully integrated "Knowledgebase".

Reliance on Data Driven Approach

The Knowledgebase design relies substantially upon a data-driven approach. Any effort to promote clear thinking about data and reduce the complexity of its relationships with other data and with rules and constraints upon the data, as a standard methodology, will be effort well spent. The data-driven approach alone can have an enormous impact on improving the quality of software, because as much as 45% of all software errors may be directly attributable to data definitions and data access operations [Rubey].

Advantage Over Process Oriented Approach

Incorporation of an effective Knowledgebase design will have a significant advantage over a traditional process oriented system-development life cycle technique. The degree of improvement depends upon the magnitude of change from a process oriented approach to the data- driven Knowledgebase approach.

Computable Specifications

The ultimate implementation of the Knowledgebase approach will have only one source document: the system specifications. When the specifications are fully accurate and complete, so too will be the system. There will be no intermediate stages. Incomplete, inconsistent, and vague specifications will be rejected. "Programming" by specifications will also allow for very fast and efficient prototyping. The specifications will be entirely computable, and will rely upon mathematics for provably correct implementation.

The DATA DRIVEN APPROACH

Data requirements are frequently vague and ambiguous, and it is often a non-trivial task to understand the ambiguities.

The data driven approach requires a complete understanding of the data. The following steps should not be considered a "cook-book" approach, but are some steps that are necessary to build a logical structure.

* Collect External (User) Views

* Identify all entities in each View

* Identify attributes of each View

* Define relationships in each View

* Normalize Entities of each External View

* Combine Normalized Entities

* Perform traffic (performance) analysis

COMPONENTS of a KNOWLEDGE BASE

Collection of Files

A Knowledge Base is very simply a collection of related files, similar to a data base except that the Knowledge Base includes all security, constraints, and rules (logic) about the data it holds.

Collection of Entities

Each file in the Knowledge Base is a collection of entities. (Note that no indication is made as to the medium upon which the file resides, it might be disc storage, but might just as well be core memory, canvas, or wet concrete).

Collections of Relations

An entity is something about which someone collects and retains information. Each entity will have one or more candidate keys, and will have one or more attributes. For example, the employee entity might collect and retain all information about employees.

Candidate Keys

A candidate key is one or more fields in an entity which together form a unique identity for the entity. If one or more candidate keys exist, it is customary to identify one as the primary key, and the others as alternate keys. If only one key exists, it is also known as the primary key. The primary key has the additional constraint that none of its component fields may be empty, null, or undefined. For example, the "employee#" might be the key of the employee entity.

Composite Keys

A Composite Key is a Key which is composed of more than one Field. For example, in an entity which keeps track of the time an employee spends on a particular DP project, we might have a composite key composed of the fields "project#" and "employee#".

Foreign Keys

A Foreign Key is a non-key field in the entity which may assume only the values of a Key in another entity, and is in fact contains the same information as a Key in that other entity. For example, if we have an entity to keep track of DP projects we would probably include the "project-mgr#" in the entity; the "project-mgr#" would also be likely to be a Key in another entity that has the manager's name. The "project- mgr#" in the first entity would then be considered a Foreign Key.

Search Fields

Note that a Key is not the a synonym for a Search Field. A Search Field is a field which is internally indexed in some fashion to simplify the access of a record based upon one or more given values.

Attributes

An attribute is a single field that describes a property of an entity at a given time. For example, "programmer-name" and "language" might be attributes of the entity "programmer".

Fields

There are several types of fields in the Knowledge Base, which contain data or information about data (edits, constraints, rules, etc.), the stored field, logical field, and virtual field.

Stored Field

A Stored Field is the smallest unit of information stored in the Knowledge Base. This is the actual internal representation of the field. For example, a binary representations of "gross-pay".

Logical Field

A Logical Field is the representation of a stored field actually presented to a user (this might involve an internal numeric to ASCII conversion, or conversion of an internal representation of an enumerated type or property). For example, internally a field may be represented as a floating point number, but the Logical Field would be the displayable ASCII conversion of the stored field's contents.

Virtual Field

A Virtual Field is a special type of logical field without a single stored counterpart, but which instead is derived from another or other fields. It may be a computation on other fields, for example. This is very similar in concept to the use of some cells in a spreadsheet application. One example might be that the employee entity contains the stored fields "gross-pay" and "deductions", it could also have a Virtual Field "net-pay" which would be defined as the difference between the stored fields "gross-pay" and "deductions".

Information Fields

Fields in the ultimate form of the Knowledge Base are not restricted to data, but are similar to a cell in one of the popular spreadsheets. For example, we could specify in the receivables entity a virtual field "total-receivables" defined as the total of all "unpaid-invoices" in the invoice entity. We could then define a field which contains a rule about the total receivables, namely if it exceeds $10,000 a report is to be generated to the company president detailing all of the receivables.

Virtual Keys

A Virtual Key, like a Virtual Field, is a key without a single stored counterpart, but is derived from one or more other Keys, or a portion of another Composite Key. It has special uses to implement the various types of views described here, and also is a method of ensuring key constraint (namely that a particular key value in one entity, or portion of a composite key, can not exist unless that same value exists within the key of another specified entity).

Virtual Entity

A Virtual Entity is an entity without a single stored counterpart, but is derived from one or more other entities. The Primary Keys from the derived entities must be combined into a Virtual Key which uniquely identifies the Virtual Entity.

Views

A Knowledge Base also makes extensive use of several different types of views; a view being a relationship that either exists in its own right or is derived from one or more relationships; it is composed of Virtual Entities. There are several types of views available in the Knowledge Base, some of the more useful are Input Views, Representational Views, and Output Views.

Input View

An Input View is a view which defines the input of information into the Knowledge Base; the source. Examples might be, a CRT screen or window for inquiring upon or modifying information, batch input, or the radiation sensing module on Three Mile Island.

Representational View

A Representational View is a particular type of View with very powerful implications. Some of the uses of a Representational View might be to combine several relationships together (conceptual simplification), filter a relationship or entity (security), copy an existing relationship --say from external storage medium to core (performance).

Fragment View

A Fragmented View is a particular type of Representational View, which contains a specified portion of one or more Views or entities. It can be particularly valuable for performance reasons. For example, we might set up several Fragmented Views of a warehouse entity, one each for London, Paris, New York, and Beirut, each containing only those records for the local entity. This could be used to benefit performance on an internationally distributed Knowledge Base.

Output View

An Output View is a view which defines the output of information from the Knowledge Base; the sink. Examples might be a report, a disc file, or a command to turn on the percolator.

LOGICAL KNOWLEDGE BASE DESIGN

Automated System Creation

One of the ultimate goals of the Knowledge Base system design is to permit one to create the entire Knowledge Base automatically from a set of specifications. Specifications for which there is sufficient detail that a computer program can validate many issues relating to correctness and completeness; specifications which are computable. The system implementation would then be automatically generated with no human intervention, guaranteeing absolute conformance with the specifications.

Certified Correctness

In order to automatically validate a system's specifications, it is necessary that there be sufficient detail and no ambiguity, to ensure correct translation. In order to accomplish this it will be necessary to define the information structures in normalized form, such that internally relational mathematics can be used to translate into correct data paths and flows.

Peculiarity of UnNormalized Data

The literature has shown us many examples of various peculiarities which can occur when manipulating data representations which are not normalized. Information which is not normalized becomes subject to various peculiarities as well, and therefore the Knowledge Base depends upon ultimate normalization of the knowledge content in order to guarantee conformance with the specifications.

Problems Implementing 4GL's

It is interesting to note here, that one of the most significant difficulties in implementing some fourth generation languages is the burdensome complexity of the un-normalized data structures.

NORMALIZATION

Benefits

By normalizing our Knowledge Base we expect to receive the following benefits:

* Better understanding of, and clearer conceptualization of data and information

* Emphasis of the creative skills of designers and analysts

* Minimization (or control) of redundancy

* Increased flexibility of the underlying internal structure, as the logical structure allows may varieties of physical structure

* Decreased development time by removing unnecessary complexity about data

* Minimization of useless (and error prone) coupling of information

* Maximization of beneficial cohesion of information

* Minimization of maintenance effort, changes are simplified due to so called "data-independence", perhaps better termed "program-independence"

* Provision of a foundation for creating the actual physical representation

* Increased user satisfaction, as the design is derived directly from the user requirements (views)

* Reduction or elimination of the potential for semantic disintegrity (simply: a query whose semantics appear proper, but which return inaccurate or invalid results)

INTRODUCTION to NORMALIZATION

Decomposition Process

The process of normalization is actually a process of decomposition, in which a structure is decomposed into two or more structures, with no loss of information, eliminating some of the problems that are inherent in the un-normalized structure.

Relational Mathematics

Normalization, as we know it today, was developed within the framework of relational mathematics, originally for the purpose of creating relational data bases. Normalization, however, has significant potential value whether the ultimate physical structure is relational, network, or hierarchy.

Simple Approach

Most of the literature, today, on normalization is unnecessarily encumbered by mathematics and or definitions, and has probably led to its lack of use as a good structuring tool. It is the intention here to demonstrate normalization in a much simpler fashion than is customary, by avoiding definitions and mathematics, and looking at normalization in terms of violations of normal forms instead of in depth discussions of the decomposition process, and finally to introduce a diagramming technique which almost makes normalization automatic.

NORMALIZATION SIMPLIFIED

Five Normal Levels

We are going to look at five levels of normalization, the first through the fifth normal form. (There are additional normal forms which are not as universally applicable as these, or whose characteristics are not as readily understood or recognizable). Fifth normal form is the highest level of normalization treated here, and first normal form is the lowest or most primitive. Each succeedingly higher level of normalization is defined to include the lower level(s), a point we will not dwell on any further.

BEFORE NORMALIZATION

Remove Attribute Anomalies

Before any attempt is made to normalize the various records of a Knowledge Base, peculiarities due to the use of the fields themselves should be removed. Some of these cases are:

* information contained redundantly in more than one field (including the concept of virtual fields)

* one field containing more than one set of properties which are multi-definitional or containing properties which are not mutually exclusive

FIRST NORMAL FORM

Repeating Fields

A record is considered to be in violation of first normal form if it contains groups of repeating fields.

For example, if we wanted to keep track of each task, and the task's estimated hours to complete by project we might have the external view as in figure 2.

Obviously, this is a perfectly acceptable user view, assuming it meets the user's requirements; however, if the data is structured this way, it is not in first normal form as the fields "task#" and "estimated-hours" are repeated several times.

Anomalies

There are several problems with the data structured as in this figure:

Inserting:

Since a variable number of fields are used, we must determine all of the tasks that have been defined for a particular project before we can insert another one. What if we exceed the arbitrary number of tasks allowed here? Exceeding the arbitrary limit here could cause a very significant effort to correct.

Deleting:

In order to delete a task# from a project it will be necessary to examine all tasks in the record, and may require significant effort to remove if it is not in the last physical position of the record.

Proper Structures

This structure can be decomposed into the the two structures of figure 3, which will eliminate the problems with the un-normalized structure.

VIOLATION
OF
FIRST NORMAL FORM

KEY							
PROJECT#	PROJECT-NAME	TASK1#	TASK1-HRS	TASK2#	TASK2-HRS	TASK3#	TASK3-HRS
1	INVENTORY	1023	12	1031	45	1069	21
2	FINANCE	1054	26	1038	28		
3	MANUFACT	1062	18				
4	RECEIVABLES	1058	13	1042	38	1019	64

fig. 2

CORRECT FIRST NORMAL FORM STRUCTURE

KEY	
PROJECT#	PROJECT-NAME
1	INVENTORY
2	FINANCE
3	MANUFACT
4	RECEIVABLES

KEY		
PROJECT#	TASK#	EST-HRS
1	1023	12
1	1031	45
1	1069	21
2	1054	26
2	1038	28
3	1062	18
4	1058	13
4	1042	38
4	1019	64

fig. 3

SECOND and THIRD NORMAL FORMS

Key Dependence	Both second and third normal form require that entities are structured such that each any every attribute is dependent upon the whole key and only the key of that entity.

SECOND NORMAL FORM

Partial Composite Keys Relations

It is a violation of second normal form when a non-prime attribute (a field not forming any portion of the key) is a property of only a portion of a composite key.

For example, if we to keep track of the hours worked by each programmer for each current project we might have the structure shown in figure 4.

This is also a perfectly legitimate external (user) view. But it does not meet the requirements of second normal form. Note that the "programmer#" and the "project#" are combined to form the key (a composite key) for the entity. While the "hours-worked" is a property of the whole key as it represents the hours worked by that programmer on that project, the "project-name" is a property of only a portion of the key: the "project#".

Anomalies

Problems with structures not in second normal form:

Inserting:

We cannot insert a project number and project name until such time as we have a programmer assigned to the project.

Deleting:

If we should remove the sole programmer from a project, we will lose the project number and project name.

VIOLATION OF SECOND NORMAL FORM

KEY

PROGRAMMER#	PROJECT#	PROJECT-NAME	HOURS-WORKED
300-51-6632	1	INVENTORY	132
300-51-6632	2	FINANCE	240
170-23-6780	2	FINANCE	240
282-82-2560	3	MANUFACT	40
276-37-8070	4	RECEIVABLES	25

fig. 4

Updating:

There is much potential redundancy in these records. A project name will need to be repeated once for each programmer assigned to that project. If a project should need to be renamed we will need to search many records to ensure all have been updated or risk inconsistency.

Proper Structures

This structure can be decomposed into the structure of figure 5 which will eliminate the problems with the un-normalized structure.

CORRECT SECOND NORMAL FORM STRUCTURE

KEY	
PROJECT#	PROJECT-NAME
1	INVENTORY
2	FINANCE
3	MANUFACT
4	RECEIVABLES

KEY		
PROGRAMMER#	PROJECT#	HOURS-WORKED
300-51-6632	1	132
300-51-6623	2	240
170-23-6780	2	240
282-82-2560	3	40
276-37-8070	4	25

fig. 5

THIRD NORMAL FORM

Non-Prime Relations

It is a violation of third normal form if any non-prime attribute (an attribute not forming part of the key) is a property of any other non- prime attribute.

For example, if we have the requirement to show all projects, their names, estimated completion date, the project manager, and the manager's name we might have the representation in figure 6.

This also is a perfectly legitimate external or user view, but it is not in third normal form. The "manager-name" is a property of the "manager#", which is not the entity's key.

Anomalies

Problems with records not in third normal form:

Inserting:

It is not possible to enter a manager name and number relationship until the manager has been assigned to a project.

Deleting:

If a project, which is the last for a particular manager, is completed and deleted, we lose that manager's name and number.

Updating:

If a manager changes name (gets married or whatever), it will be necessary to search all records to update all occurrences of that manager's name, or risk inconsistent data.

This structure can be decomposed into the structures of figure 7 which will eliminate the problems associated with that structure.

VIOLATION
OF
THIRD NORMAL FORM

KEY

PROJECT#	PROJECT-NAME	COMPL-DATE	PROJECT-MGR#	MGR-NAME
1	INVENTORY	10/10/87	363-92-2109	JOHN SMITH
2	FINANCE	12/23/87	242-06-1132	GEORGE JONES
3	MANUFACT	02/28/87	196-32-8693	MARY WEATHER
4	RECEIVABLES	03/31/87	363-92-2109	JOHN SMITH

fig. 6

CORRECT THIRD NORMAL FORM STRUCTURE

KEY	
PROJECT-MGR#	**MGR-NAME**
363-92-2109	**JOHN SMITH**
242-86-1132	**GEORGE JONES**
186-32-8693	**MARY WEATHER**

KEY			
PROJECT#	**PROJECT-NAME**	**COMPL-DATE**	**PROJECT-MGR#**
1	**INVENTORY**	**10/10/87**	**363-92-2109**
2	**FINANCE**	**12/23/87**	**242-86-1132**
3	**MANUFACT**	**02/28/87**	**186-32-8693**
4	**RECEIVABLES**	**03/31/87**	**363-92-2109**

fig. 7

FOURTH and FIFTH NORMAL FORMS

Multi Valued Composite Keys	It is a violation of fourth and/or fifth normal form if an entity's key contains more than one potentially multi-valued property of that entity. (Fourth and fifth normal form differ only in decomposition to ensure non- loss decomposition).

FOURTH NORMAL FORM

If we wished to show each programmer and his skills both in data base management systems and programming languages we might have the representation in figure 8.

This also is a perfectly legitimate user view, but it is not in fourth normal form. A given programmer might have one data base skill, three language skills, another might have two data base skills, and eight language skills, etc.

Anomalies

Problems with structures not in fourth normal form:

Inserting:

Several problems arise with violations of fourth normal form, the problem is really one of permutations and combinations.

Inserting a new DBMS skill for a programmer would require searching for all of a programmer's language skills, and inserting a new record for each language skill and the new DBMS combination.

Inserting a new language skill for a programmer would require searching for all of a programmer's DBMS skills, and inserting a new record for each DBMS skill and the new language combination.

Updating:

If a vendor changes the name of their data base management system (e.g. from IMAGE to Turbo-IMAGE) it will be necessary to search many records to ensure that each occurrence is updated, or risk inconsistency.

This structure can be decomposed into the two structures of figure 9, which will eliminate the problems associated with the un-normalized structure.

VIOLATION OF FOURTH NORMAL FORM

KEY

PROGRAMMER#	DBMS	LANGUAGE
300-51-6632	IMAGE	FORTRAN
300-51-6632	IMAGE	COBOL
300-51-6632	HP/SQL	FORTRAN
300-51-6632	HP/SQL	COBOL
170-23-6780	DB2	COBOL
170-23-6780	IMAGE	COBOL
170-23-6780	DB2	RPG
170-23-6780	IMAGE	RPG

fig. 8

CORRECT FOURTH NORMAL FORM STRUCTURE

KEY

PROGRAMMER#	DBMS
300-51-6632	IMAGE
300-51-6632	HP/SQL
170-23-6780	DB2
170-23-6780	IMAGE

KEY

PROGRAMMER#	LANGUAGE
300-51-6632	FORTRAN
300-51-6632	COBOL
170-23-6780	RPG
170-23-6780	COBOL

fig. 9

FIFTH NORMAL FORM

If we were to reflect the policy that any programmer who has a skill in a particular language, and has experience on a particular vendor's equipment, that programmer will be considered to be capable of programming that language on that particular vendor's hardware (assuming that language is available on that vendor's hardware) then we might have the the external view represented in figure 10.

While this is certainly acceptable as a user view, it does not meet the requirements of fifth normal form.

Anomalies

Problems with structures not in fifth normal form:

Inserting:

When a programmer gains a skill in a particular language, it will be necessary to insert that fact once for each vendor that supports that language.

Deleting:

If a programmer who is the only programmer with a skill in that language leaves, removing his record will cause us to lose the information that one or more vendor's support that language.

If a programmer who is the only programmer who has experience with a particular vendor leaves, removing his record will cause us to lose the information that this vendor supports one or more programming languages.

Updating:

If a vendor changes its name or is acquired by another vendor, and therefore changes its name, we will need to search many records to be sure all are updated to reflect the new name, or risk inconsistent information.

This structure can be decomposed into the three structure of figure 11, which will eliminate the problems associated with the un-normalized structure.

Note that the only difference between this example of the fifth normal form case and the previous fourth normal form example is the rule about the data that we established at the beginning which causes all components of the composite key to be inter-related to each other.

VIOLATION OF FIFTH NORMAL FORM

KEY

PROGRAMMER#	HARDWARE	LANGUAGE
300-51-6632	HP	FORTRAN
300-51-6632	HP	COBOL
300-51-6632	IBM	FORTRAN
300-51-6632	IBM	COBOL
300-51-6632	HP	SPL
170-23-6780	HP	COBOL
170-23-6780	IBM	COBOL
170-23-6780	HP	RPG
170-23-6780	IBM	RPG
170-23-6780	DEC	COBOL
170-23-6780	DEC	RPG
170-23-6780	DEC	C
170-23-6780	IBM	C

fig. 10

CORRECT FIFTH NORMAL FORM STRUCTURE

KEY	
PROGRAMMER#	**HARDWARE**
300-51-6632	HP
300-51-6632	IBM
170-23-6700	HP
170-23-6700	DEC
170-23-6700	IBM

KEY	
PROGRAMMER#	**LANGUAGE**
300-51-6632	FORTRAN
300-51-6632	COBOL
300-51-6632	SPL
170-23-6700	COBOL
170-23-6700	RPG
170-23-6700	C

KEY	
HARDWARE	**LANGUAGE**
HP	FORTRAN
HP	COBOL
HP	RPG
HP	SPL
IBM	FORTRAN
IBM	COBOL
IBM	C
DEC	COBOL
DEC	RPG
DEC	FORTRAN
DEC	C

fig. 11

A SIMPLER APPROACH TO NORMALIZATION

Diagramming Techniques

As is said, "a picture is worth a thousand words". The task of normalizing logical information structures is immensely simplified by the use of diagrams. With appropriate diagramming techniques, it is possible to ensure normalization during the diagramming process, and finish with a document which lends itself very well to the implementation of the physical structure.

Sketching Relations

The diagramming technique we will use here, adapted in part from several other common techniques, will represent the fields of an entity, each enclosed within a block. The association between the fields in the entity will be indicated with a line drawn between them. Adjacent to the entities connected by the line, and drawn on top of the line will be one or more of the following:

* a single cross line, representing an association of one

* a circle, representing the possibility of no association

* a crow's foot, representing a association of more than one

For example, we could represent the employee entity as in figure 12.

Note the single cross line adjacent to the "employee#" and the "employee-name". This indicates a one to one association.

In this example the "employee#" is the key, and therefore the usual box around the field is also inscribed by another box, which indicates that this field is the key.

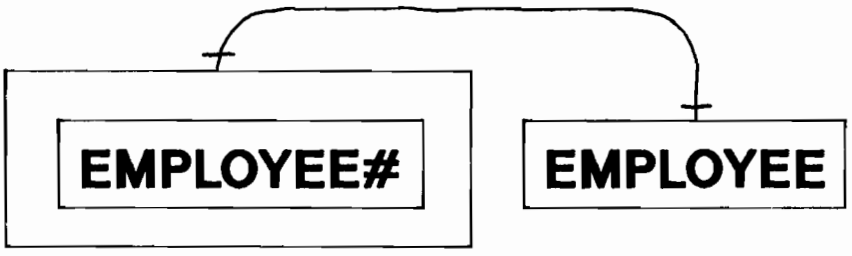


fig. 12

FIRST NORMAL FORM

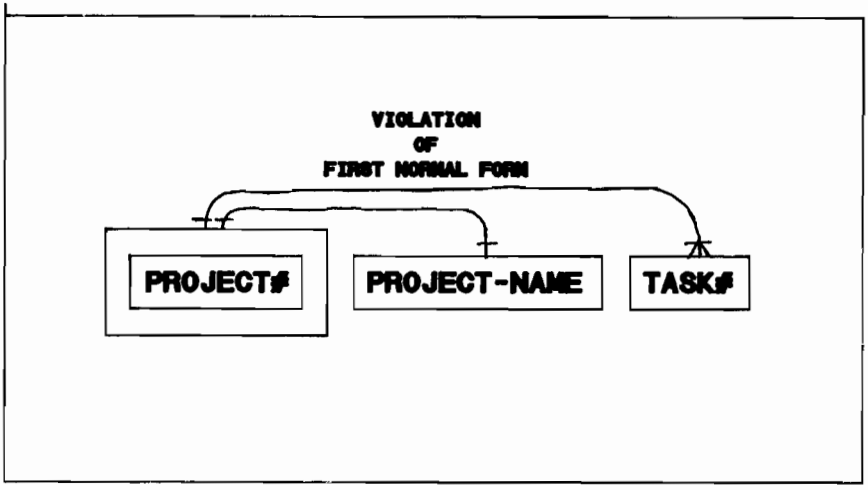
Violation

In figure 13 we represent the diagram of our first normal form example. The crow's feet immediately show us that there is a one to many relationship from the "project#" to the "task#". This diagram show the violation of first normal form.

Correct

If we decompose this structure into a permissible first normal form structure we have the corrected from also in figure 13.

Note that there are now no cases of one to many relationships between one key and its associated attributes within an entity. However, there is a one to many relationship between the key in one of the relationships and a portion of the key in the other. Note that the "project#" fields are each foreign keys as well as a key or portion of a composite key in their own entity. Foreign keys usually indicate the need for key constraint (in this case we would not wish to allow an entry for a project and task if there is no project entry). Foreign keys are frequently desirable as search fields in the final physical structure.


VIOLATION
OF
FIRST NORMAL FORM
PROJECT#
PROJECT-NAME
TASK#

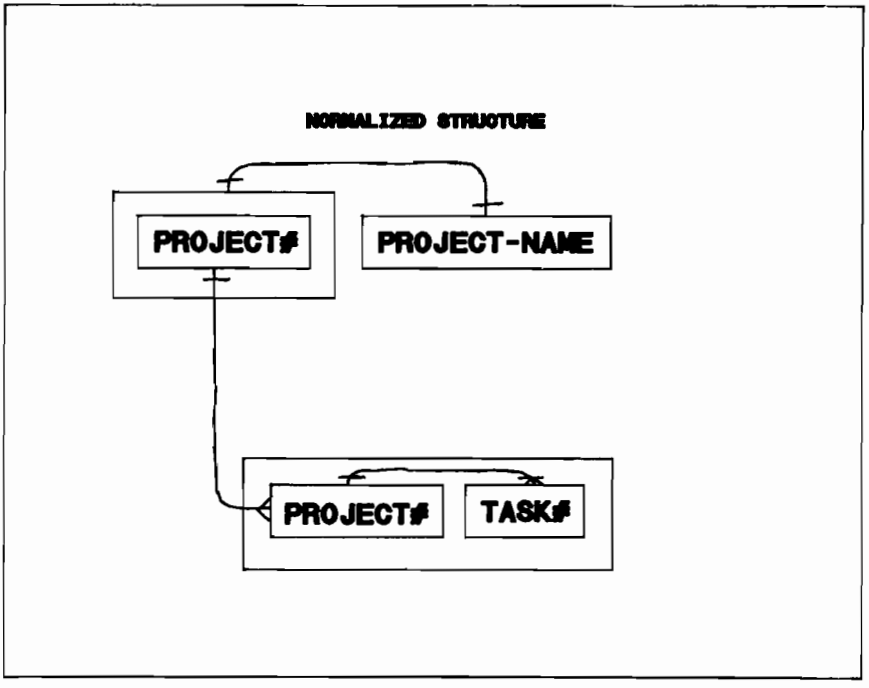

NORMALIZED STRUCTURE
PROJECT#
PROJECT-NAME
PROJECT#
TASK#

fig. 13

SECOND NORMAL FORM

Violation

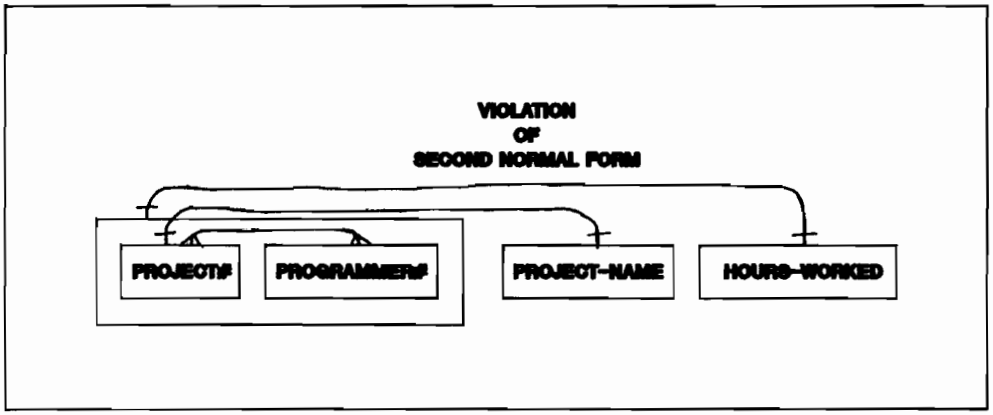
Figure 14 represents a diagram showing a violation of the second normal form structure.

Note that since the "project#" and the "programmer" are together enclosed in a box, they together form a composite key. The diagram very clearly shows that while "hours-worked" is dependent upon the whole key, the "project-name" clearly is not.

Corrected

Figure 14 also shows the corrected structures that satisfy second normal form.

In this case all attributes of all entities are dependent upon the whole key of that entity. Note again that again there is a one to many relationship between the key in one entity and a portion of the key in another entity, and again we have the existence of foreign keys.


VIOLATION
OF
SECOND NORMAL FORM
PROJECT#
PROGRAMMER#
PROJECT-NAME
HOURS-WORKED

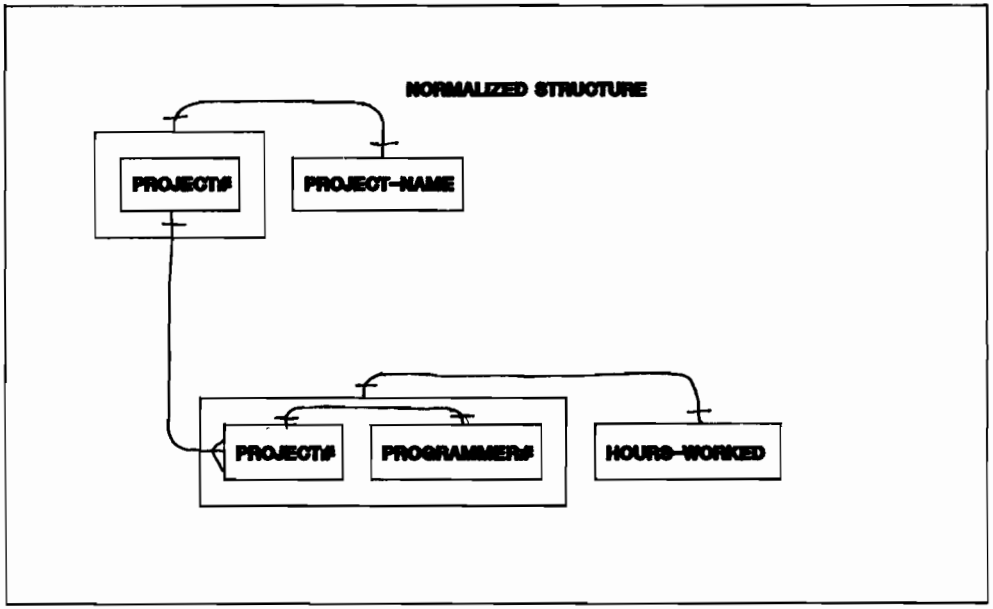

NORMALIZED STRUCTURE
PROJECT#
PROJECT-NAME
PROJECT#
PROGRAMMER#
HOURS-WORKED

fig. 14

THIRD NORMAL FORM

Violation

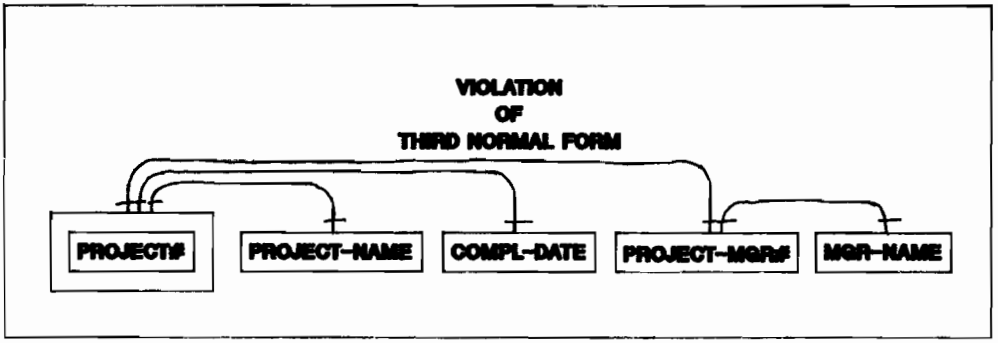
Figure 15 represents a diagram which contains a structure in violation of third normal form.

This diagram clearly shows that not all attributes in the relationship depend upon the key. The "project#" is the key of the entity, but the "manager-name" depends upon another attribute.

Correct

Figure 15 also show a diagram with structure that represent the same data but satisfy third normal form.

Note in this diagram that all attributes in each relationship depend solely upon the key to that entity. Note also here that there is a many to one relationship between the "project-mgr#" of the project entity and the "project-mgr#" of the project manager entity. In this case we have a field which is non-prime (non-key) in the project entity, but yet which is also a foreign key as it exists in the employee entity.


VIOLATION
OF
THIRD NORMAL FORM
PROJECT#
PROJECT-NAME
COMPL-DATE
PROJECT-MGR#
MGR-NAME

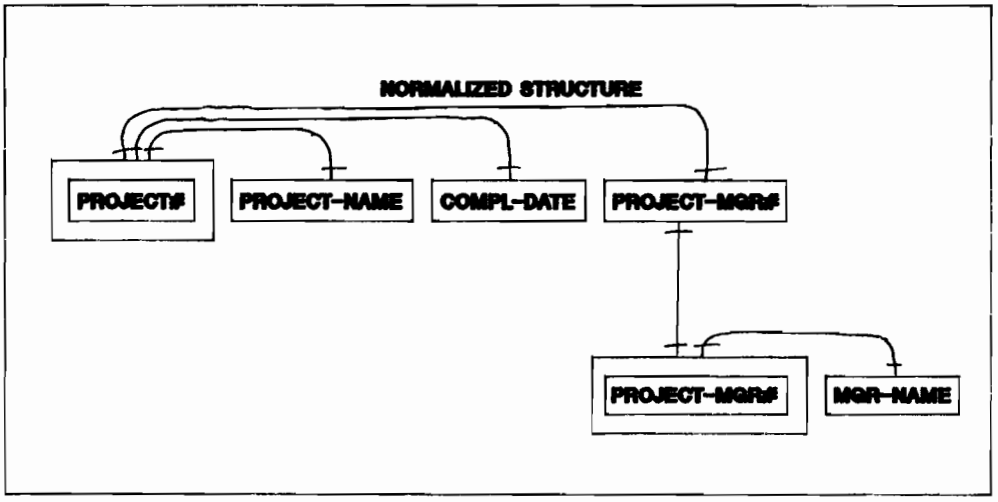

NORMALIZED STRUCTURE
PROJECT#
PROJECT-NAME
COMPL-DATE
PROJECT-MGR#
PROJECT-MGR#
MGR-NAME

fig. 15

FOURTH NORMAL FORM

Violation

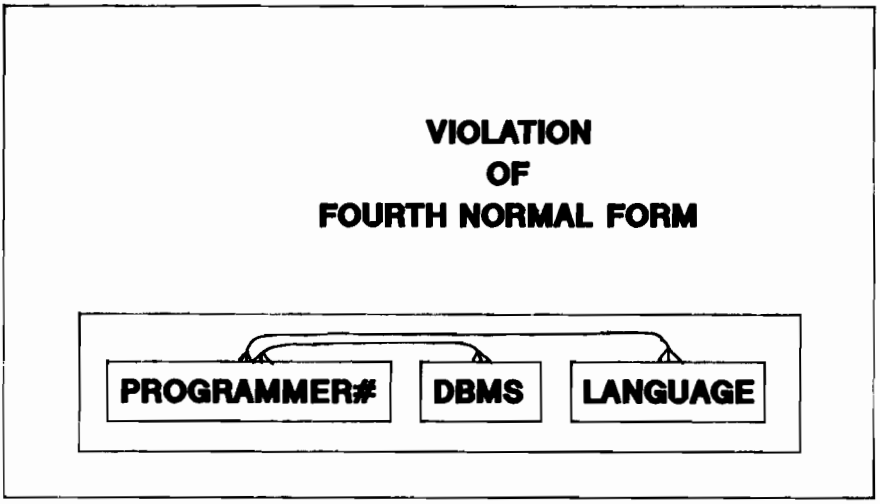
Figure 16 shows a diagram that is in violation of fourth normal form.

As can be clearly seen, the key is made up of more than two fields all of which have more than one to one associations with each other, and that is simply the violation of fourth normal form.

Correct

Figure 16 also represents a diagram which satisfies the requirements of fourth normal form.

Note that the correct version does not have any composite keys fields comprised of more than two fields with many to many relationships between them.


VIOLATION
OF
FOURTH NORMAL FORM
PROGRAMMER#
DBMS
LANGUAGE

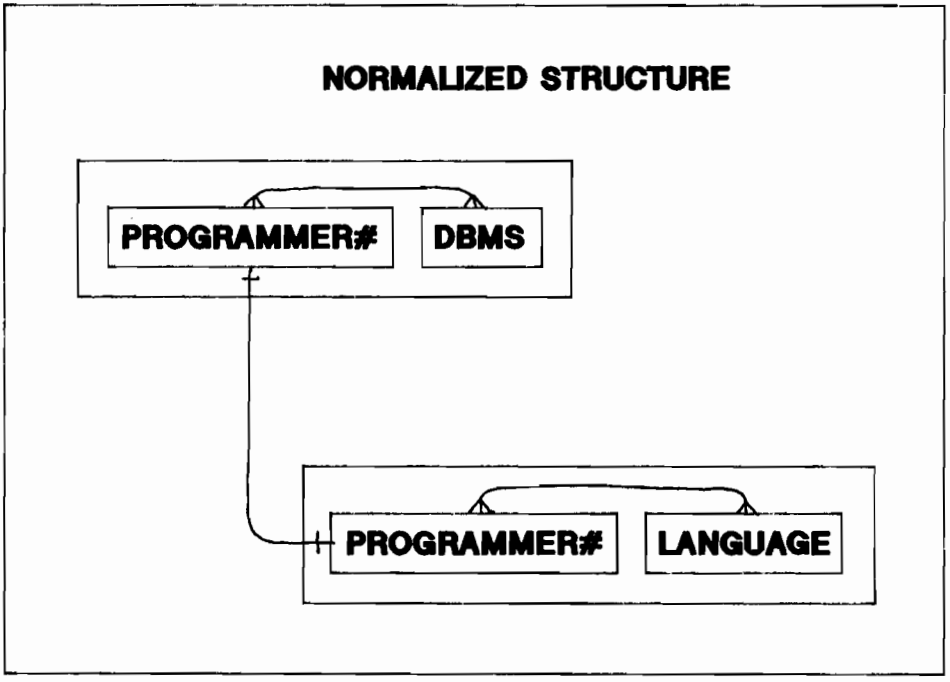

NORMALIZED STRUCTURE
PROGRAMMER#
DBMS
PROGRAMMER#
LANGUAGE

fig. 16

FIFTH NORMAL FORM

Violation

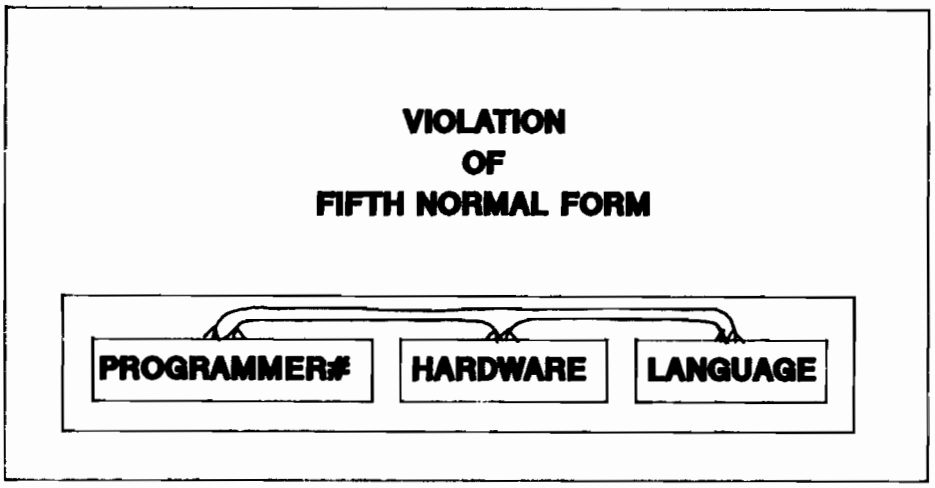
Figure 17 represents a diagram of a structure that is not if fifth normal form.

Note that this diagram also shows a composite key composed of two or more fields which are associated with other than one to one relationships among them. This diagram is in violation of fifth normal form, and additionally does not adequately describe the rules we have placed upon the data.

Correct

Figure 17 also shows a diagram of a structure which satisfies the requirements of fifth normal form.

Note that the relationship between fields composing the keys are potentially many to many, but there are no such cases involving composite keys formed by more than two fields.


VIOLATION
OF
FIFTH NORMAL FORM
PROGRAMMER#
HARDWARE
LANGUAGE

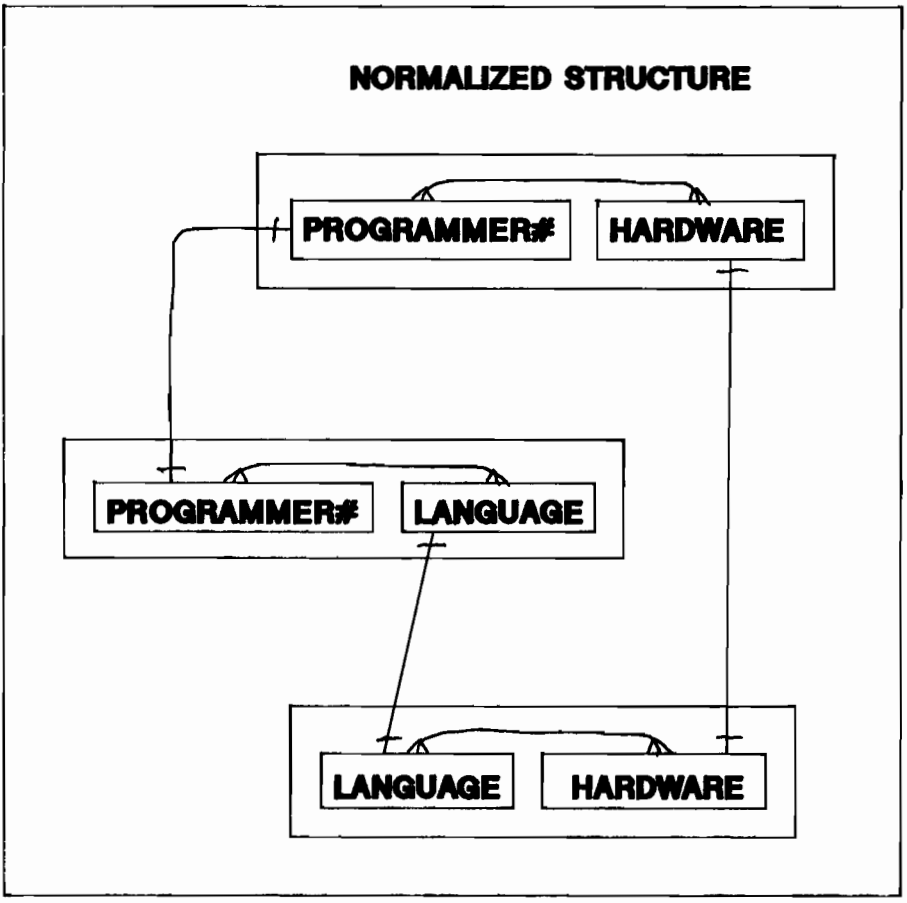

NORMALIZED STRUCTURE
PROGRAMMER#
HARDWARE
PROGRAMMER#
LANGUAGE
LANGUAGE
HARDWARE

fig. 17

PHYSICAL KNOWLEDGE BASE DESIGN

Once we have a normalized logical Knowledge Base (or data base) design, it becomes almost a trivial matter to reduce it to one of the possible physical designs available in a relational, hierarchical, or network structure.

Combine Normalized Views

When we have combined all of the normalized structures from the preceding exercise, we arrive at a diagram which look like figure 18.

Hierarchy Physical Structure

To implement this logical structure in a hierarchical physical structure we begin by looking for all of the highest level segments. In this diagram there are actually two root segments: the project entity (project#, project-name, etc) and the employee entity (employee#, employee), consequently this logical structure will be implemented by way of two separate physical hierarchical data bases. Each of the lower levels will become subordinate segments to the respective root segments.

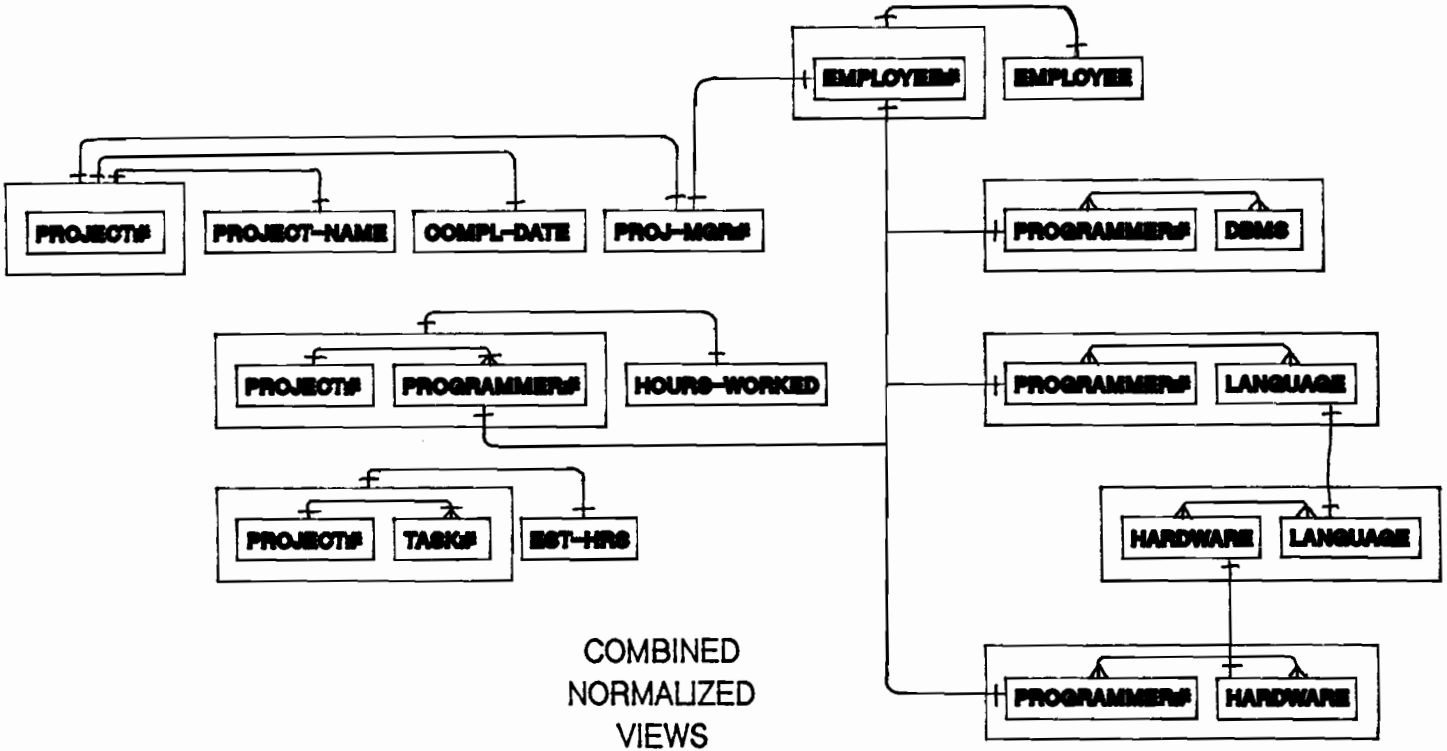

EMPLOYEE#
EMPLOYEE
PROJECT#
PROJECT-NAME
COMPL-DATE
PROJ-MGR#
PROGRAMMER#
DBMS
PROJECT#
PROGRAMMER#
HOURS-WORKED
PROGRAMMER#
LANGUAGE
PROJECT#
TASK#
EST-HRS
HARDWARE
LANGUAGE
PROGRAMMER#
HARDWARE
COMBINED
NORMALIZED
VIEWS

fig. 18

Network and Relational Structures

To implement this logical structure in a network or relational physical structure is very much the same. Each entity will be either a separate data set in the network structure, or a separate table in the relational structure.

Each case where we have drawn a link between one entity and another is a candidate for being defined as a search item (most will usually need to be a search item), as they represent foreign keys. There may be other cases where it is desirable to add an index or search item, for convenient access.

The choice of which fields are to be used as search items unfortunately must depend to a large degree upon judgment of performance requirements, and the specification of search items might be the deciding factor over what type of data set is used in a network structure, and whether or not to define indices in a relational structure.

Assuming that we desire to have all foreign keys defined as search fields (which very frequently is the case), we can easily convert this logical structure into the IMAGE and relational physical structures of figure 19 and 20.

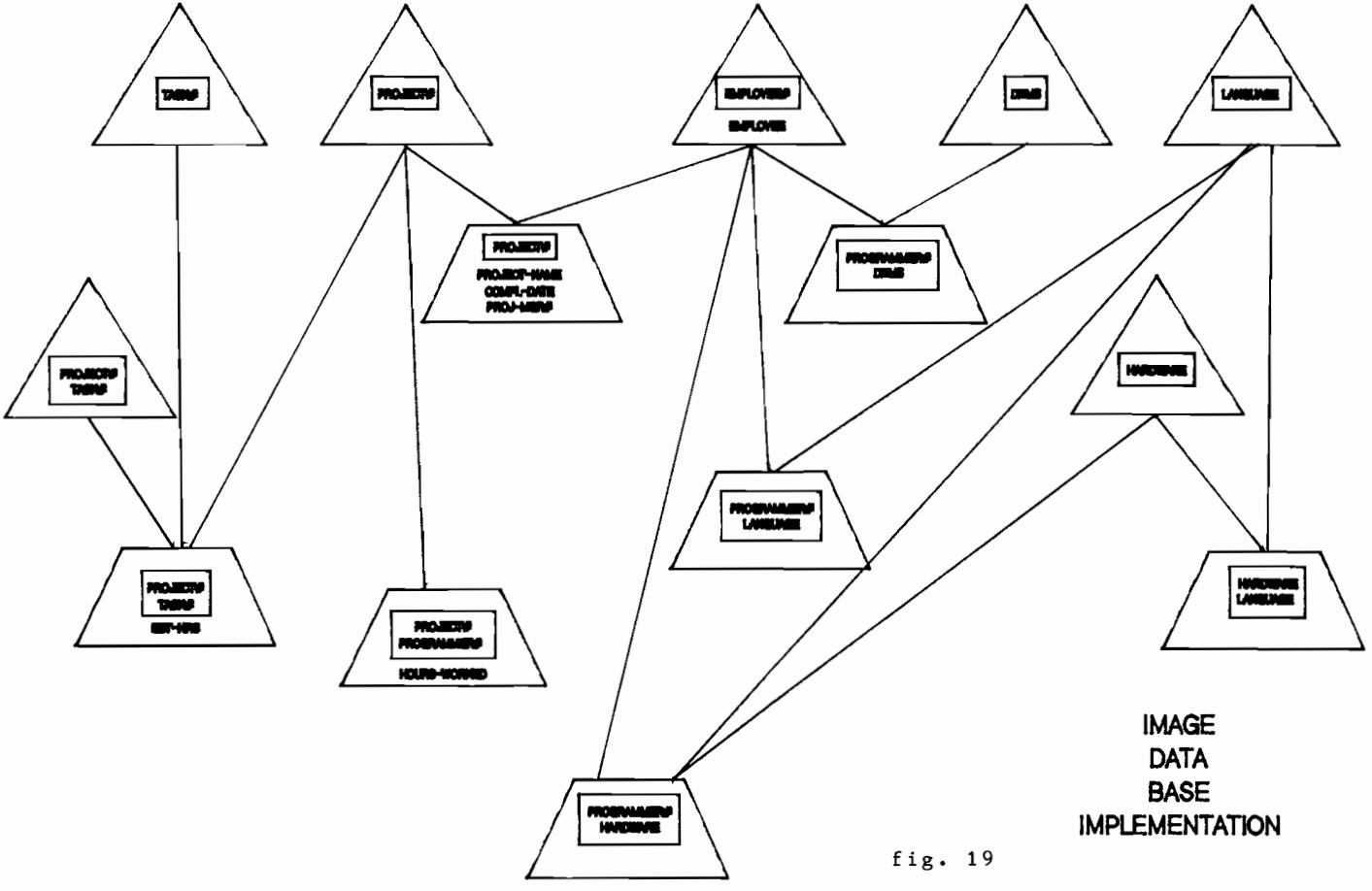


IMAGE
DATA
BASE
IMPLEMENTATION

fig. 19

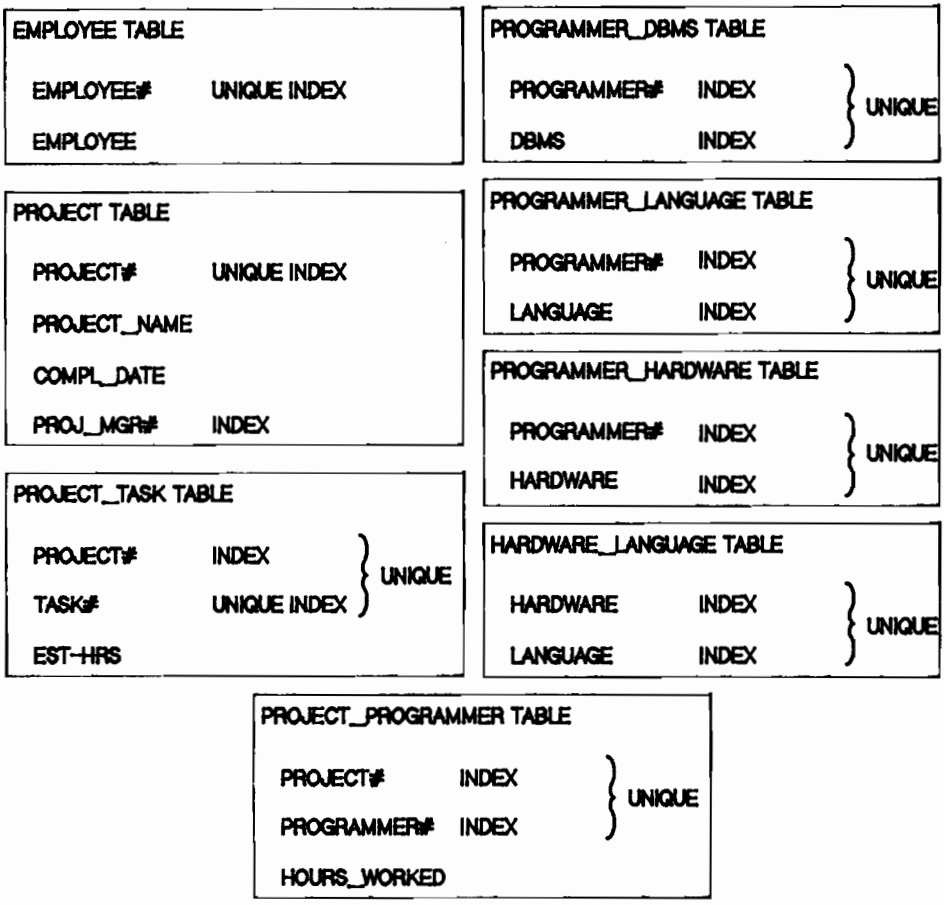


fig. 20

RELATIONAL DATA BASE IMPLEMENTATION

Yet FURTHER SIMPLIFICATION

Interactive Diagramming Tools

The ultimate implementation of the Knowledgebase will provide diagramming tools that will permit a designer or analyst to interactively specify the entities of the system he or she is designing. Using this diagramming technique, the Knowledgebase system will not permit a design which is not ultimately normalized. In other words, the graphics form a window into the Knowledgebase which ensure that the specification of logical structures are mathematically proper and computable.

Additionally, this same window will be used to define all constraints upon all fields of all entities. This will be the only place where that will be necessary (or possible).

Mapping External Views

All external (user) views will be represented here also, such as input screens and output reports. The fields on the external views will be graphically mapped to the corresponding fields in the defined entities. Constraints upon fields in input views may be defined as more restrictive (but never less) than their mapped counterparts in the stored entities, otherwise they will inherit the constraints of their mapped counterparts.

User Requirements

Users will be able to easily lay out their requirements for input views and output views using this graphical interface. The analyst or designer will then map these views to the underlying entities, and in short show the data flow. When this task is complete, the specifications will be both complete and computable, or rejected by the Knowledgebase. The complete specifications will enable the Knowledgebase to generate complete, sound, and error free software, entirely correct with respect to the specifications.

Resolving Specification Problems

Users and analysts or designers will be able to work more closely together to develop systems and resolve problems with the specifications and requirements. Prototyping will be immensely simplified, helping to provide immediate and concrete feedback to the users.

The analyst or designer will be able to graphically select from the various types of views available to the Knowledge base to simplify portions of the system, implement security, or improve performance, etc.

System Performance Hooks

An interesting sidelight worth mentioning here is that, as mentioned earlier, the files of the Knowledgebase are not required to reside on an external storage medium; not only does this provide powerful potentials for improving performance (such as declaring that heavily trafficked entities reside in main memory as well as on disc), but also tremendously expands the utility of the Knowledgebase concept.

Operating System as Knowledge Base

It would be entirely possible to implement most portions of an operating system entirely within the scope of a Knowledgebase. The system tables would be declared as files residing in high speed memory. The overall design of the system would be so well understood by the system, that advanced methods of eliminating deadlocks and reducing predicated locking times and queuing could now be moved from the blue sky of academia to reality.

Embedded Systems as Knowledge Base

The Knowledgebase implementation will also lend itself very well to embedded computer systems or process control. Various sensors that may be utilized by a process control system are simply another type of user input view. The various controllers are just another type of output view.

Artificial Intelligence Application

And last but not least, the Knowledgebase system will function very well in an artificial intelligence environment. In fact, a portion of this concept has been derived directly from some of the current state of artificial intelligence, particularly the incorporation of both facts (data) and rules (logic) into the same logical structure. This will simplify a ProLog-like process being able to derive conclusions from the Knowledgebase. The organization of the Knowledgebase (especially normalization of facts and rules) will likely advance the time when artificial intelligence systems become commercially more attractive and feasible.

The FUTURE IS NOW

Implementing Today

The ultimate implementation of this Knowledgebase concept does not yet exist in its entirety, but that should not stop us from implementing our own, to the degree possible.

Manual Enforcement

Much of what is involved is to reduce the complexities and redundancies of existing systems, while trying to approach a methodology that is precisely correct. What is not available now is a system that rejects imprecision and unnecessary complexity, but that can be enforced by the usual methods of enforcement and discipline.

Existing Components

Beginning with a good data dictionary to define all attributes of a system, a data base could be automatically derived (based upon the normalized structures defined in the dictionary), and all of the attributes used within the various programs could be derived (such as COBOL copylibs).

Modular Code

The logic to access each unique view within the system would be coded and placed into a shareable library (of executable or source code), beginning with the most primitive. The logic for each view that contained other more primitive views would access that more primitive view by its library routine. In short, no code would be permitted to be duplicated, anywhere or for any reason. Each bit of data would have one and only one primitive route into the system, and one and only one primitive route out of the system by way of the library routines designed for enabling the respective views. In this way (and only this way) will ultimate code and data independence be achieved. This ensures that for each atomic change to the system, there will be one and only one module where a change needs to be made (although more than one module may need recompiling).

REFERENCES

Lewis — Lewis, Charles J., "Understanding Database and Data Base", Journal of Systems Management, Cleveland, Oh, Sept 1977.

Martin A — Martin, James, "System Design From Provably Correct Constructs", Prentice-Hall, Englewood Cliffs, NJ, 1985, p40.

Boehm A — Boehm, B. W., "Software Engineering: Research and Design Trends and Defense Needs", Research Directions in Software Technology, Mit Press, Cambridge, MA, 1979, p44-86.

Melcher — Melcher, Daniel, "Automation: Rosey Prospects and Cold Facts", Library Journal, March 15, 1968, p1105.

Heller — Heller, Robert, "The Great Executive Dream", Delacorte Press, New York, NY, 1972, p226.

Naur — Naur, P., and Randell, B., eds., "Software Engineering: Report on a Conference Sponsored by the NATO Science Committee", NATO Scientific Affairs Division, Brussels, Belgium, 1968, p121.

Lecht — Lecht, C. P., "The Waves of Change", McGraw-Hill, New York, NY, 1977.

De Roze — De Roze, B. C., and Numan, T. H., "The Software Life Cycle - A Management and Technological Challenge in the Department of Defense", IEEE Transactions on Software Engineering, July 1978, p308-318.

Boehm B — Boehm, B. W., "Software and Its Impact: A Quantitative Assessment", Datamation, May 1973, p48-59.

Thayer — Thayer, R. H., "Rome Air Development Center Research and Design in Computer Language Controls and Software Engineering Techniques", RADC-TR-74-80, Griffis Air Force Base, Rome, NY, 1974.

Ulsamer — Ulsamer, E., "Computers - Key to Tomorrow's Air Force", Air Force Magazine, July 1973, p46-52.

Martin B — Martin, James, "System Design From Provably Correct Constructs", Prentice-Hall, Englewood Cliffs, NJ, 1985, p5.

Campbell A — Campbell, Donald F., "Reducing Applications Backlog with 4GLs", the Handbook of MIS Management, Supplement I, Auerbach Publishers, Inc., Boston, MA, 1986, p1091.

Campbell B — Campbell, Donald F., "Reducing Applications Backlog with 4GLs", the Handbook of MIS Management, Supplement I, Auerbach Publishers, Inc., Boston, MA, 1986, p1095.

Rubey — Rubey, R. J., et al, "Comparative Evaluations of PL/I", ESD-TR-68-150, U.S. Air Force, Bedford, MA, 1968.

DESIGN FEATURES OF THE MPE XL USER INTERFACE

by Jeff Vance & John Korondy
Hewlett-Packard
19447 Pruneridge Avenue
Cupertino, California 95014

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. 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 architectual 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 acheived 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. file, 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. Three major differences exist 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/program files. 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 be 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 jcws, 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 jcws.

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 occurence 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 hpresult+10
```

As mentioned 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 dereferencing 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.

Implicit 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 arithemetic 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.

It is also possible to use expressions in any MPE XL command by explicitly dereferencing the expression, referred to as "expression substitution". To do this, enclose the expression within square brackets and precede it with an exclamation mark: ![expression]. For example, :print ![prefix+'01'+suffix], where the suffix and prefix user-defined variables contain string values.

Mixed expressions are not allowed, for example, calc "a"+1 would result in an error. Standard precedence rules apply, with variable dereferencing superceding 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}
{ assume file "BBAA" exists }
:calc finfo(b,0)                        {TRUE}
{ assume eof on "SL" is 5540 }
:file x;disc=![100+finfo('sl',19)/2]    {:file x;disc=2820}
{ assume HEAPSIZE=500 }
:run pgm;nmheap=![heapsize*4]           {:run pgm;nmheap=2000}
```

E. Command File Examples.

The first command file is more fully commented than the others and should be read throughly.

1. "CIERR" - prints the error message text associated with either the current value of the CIERROR jcw, or a passed cierror value.

```
PARM cierr=0
OPTION nolist
COMMENT  Parameter cierr is not required. Its default value is 0.
COMMENT  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. Note that parameters
COMMENT  must always be explicitly dereferenced. Implicit
COMMENT  dereferencing is only available to global variables.
if !cierr <> 0 then
   comment  Since we're dealing with global variables,
   comment  use an unlikely name.
   setvar _cierr cierror
   setjcw cierror !cierr
   endif
COMMENT  Retrieve the error text associated to the value of CIERROR.
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
  COMMENT  and resetting CIERROR if necessary.
  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 command file.
```

Usage:

```
  :tail catalog.pub.sys
  :tail myletter 40
```

3. "PRT" - this command file prints to EPOC up to 6 files.

```
  PARM f1,f2='',f3='',f4='',f5='',f6='';env=elite
  COMMENT  Prints f1-f6 to EPOC. User may specify environment.
  setvar i 1
  while '!"f!i"' <> ''
    if finfo('!"f!i"',0) then
```

```
        file !"f!i";dev=epoc;env=!env.hpenv.sys
        echo (PRT): Printing of ![finfo('!"f!i"',1)] in progress...
        print !"f!i",*!"f!i"
        reset !"f!i"
     else
        echo (PRT): ![ups('!"f!i"')] not found...printing skipped.
     endif
     setvar i i+1
  endwhile
  deletevar i
  COMMENT  End of command file.
```

Usage: (assume logon group=LEI, logon account=UI and file MEMO does not exist)

```
  :prt letter,memo,paper.doc
  (PRT): Printing of LETTER.LEI.UI in progress...
  (PRT): MEMO not found...printing skipped.
  (PRT): Printing of PAPER.DOC.UI in progress...
```

4. "ME" - this command file provides a convenient :showme format.

```
  PARM flag=''
  COMMENT  Shows user environment info similar to :showme.
  COMMENT  If "flag" is non-null then the user's
  COMMENT  capabilities are also displayed.
  if hpinbreak then
     echo (#!hpjobtype!hpjobnum)!hpuser.!hpaccount,!hpgroup&
          Ldev=!hpldevin  CI=hpcidepth  "!hpsysname"&
          !hptimef    <In Break>
  else
     echo (#!hpjobtype!hpjobnum)!hpuser.!hpaccount,!hpgroup&
          Ldev=!hpldevin  CI=hpcidepth  "!hpsysname"&
          !hptimef
  endif
  if "!flag" <> "" then
     echo !hpusercapf
  endif
  COMMENT  End of command file.
```

Usage:

```
  :me
  (#S97)JEFF.UI,CI  Ldev=210  CI=1  "JOY"   5:17 PM
  :me x
  (#S6)MANAGER.SYS,SCRIPTS  Ldev=20  CI=1  "JOY"   10:30 AM   <In Break>
  AM,AL,GL,DI,CV,UV,LG,CS,ND,SF,IA,BA,PH,DS,MR,PM
```

5. "ADDCAP" - this command file adds a capability to the logon user's current capabilities.

```
PARM cap
COMMENT  Syntax: addcap <capability>
COMMENT  Adds "cap" to the current user's existing capabilities. 'AM' is
COMMENT  required to execute the :altuser command. The new capability
COMMENT  takes effect only after the user re-logson.
comment: make sure that the user doesn't already have "cap"
if pos(ups('!cap'),hpusercapf) > 0 then
   echo (ADDCAP): You already have ![ups('!cap')].
else
   setvar cierror 0
   continue
   altuser !hpuser;cap=![hpusercapf+',!cap']
   if cierror <> 0 then
      echo (ADDCAP): Your capabilities remain: !hpusercapf.
   else
      echo (ADDCAP): Your new capabilities are: &
           ![hpusercapf+','+ups('!cap')]
      setvar addcap_temp 'N'
      continue
      input addcap_temp,'(ADDCAP): Log off/on now (Y/N)> ',10
      if cierror = 9003 then
         echo (ADDCAP): Timed 10-second read expired. Logon cancelled.
      else
         if ups(lft(addcap_temp,1)) <> 'Y' then
            echo (ADDCAP): New capabilities take effect at next logon.
         else
            hello !hpjobname,!hpuser.!hpaccount,!hpgroup
         endif
      deletevar addcap_temp
      endif
   endif
endif

Usage:

:listuser foo
********************
USER: FOO.UI

HOME GROUP: CI                    PASSWORD: **
MAX PRI   : 150                   LOC ATTR: $00000000
LOGON CNT : 1
CAP: AM,ND,SF,BA,IA,PM,MR,PH

:addcap pm
(ADDCAP): You already have PM.

:addcap al
(ADDCAP): Your new capabilities are: AM,ND,SF,BA,IA,PM,MR,PH,AL
(ADDCAP): Log off/on now (Y/N)> no
(ADDCAP): New capabilities take effect at next logon.

:addcap al
(ADDCAP): Your new capabilities are: AM,ND,SF,BA,IA,PM,MR,PH,AL
```

```
(ADDCAP): Log off/on now (Y/N)> yes
CPU=10. CONNECT=71. WED, MAY  6, 1987, 12:44 PM.
HP3000 / MPE XL  9.20.22.  WED, MAY  6, 1987, 12:45 PM.
============================================================
```

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.

```
PARM size=""
COMMENT  1) echos current size of redo stack, or 2) adjusts
COMMENT  hpredosize to hpredosize+"size", or 3) sets hpredosize
COMMENT  to "size".
COMMENT
if "!size" = "" then
   echo (RSIZE): The redo stack size is !hpredosize.
else
   if typeof(!size) <> 1 then
      comment  Not numeric.
      echo (RSIZE): If parameter is supplied it must be a signed or &
           unassigned number.
   else
      setvar cierror 0
      if (lft("!size",1) = "+") or (lft("!size",1) = "-") then
         continue
         setvar hpredosize ![hpredosize+!size]
         if cierror = 0 then
            echo (RSIZE): The redo stack is now !hpredosize.
         endif
      else
         continue
         setvar hpredosize !size
      endif
      if cierror <> 0 then
         echo (RSIZE): The redo stack remains !hpredosize.
      endif
   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".

```
PARM lognum=0,mode='QUIET'
COMMENT  Purge all log####.pub.sys starting at "lognum". Echo purged
COMMENT  file if not "quiet".
setvar _logcnt 0
setvar _logx !lognum
setvar _quiet ups(lft(!mode,1)) = 'Q'
setvar _cont hpautocont
setvar _msg hpmsgfence
setvar hpautocont true
setvar hpmsgfence 2
setvar cierror 0
COMMENT  Find first log file to purge.
setvar _logname "LOG"+str('0000',1,4-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',1,4-len('!_logx'))+"!_logx"
   setvar 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.
if not(finfo('!file',0)) then
```

```
   comment  File does not exist.
   if lft('!file',1) <> '*' and lft('!file',1) <> '$' then
      comment  Qualify file before reporting non-existence.
      if pos('.','!file') > 0 then
         if pos('.',rht('!file',len('!file')-pos('.','!file'))) > 0 then
            echo ![ups('!file')] does not exist.
         else
            echo ![ups('!file')].!hpaccount does not exist.
         endif
      else
         echo ![ups('!file')].!hpgroup.!hpaccount does not exist.
      endif
   else
      echo !file does not exist.
   endif
else
   comment ** formal file designator **
   echo (FINFO): Full file description for ![finfo('!file',1)] follows:
   comment ** creator and create/modify dates **
   echo    Created by ![finfo('!file',4)] on ![finfo('!file',6)].
   echo    Modified on ![finfo('!file',8)] at ![finfo('!file',24)].
   comment ** file code **
   if finfo('!file',9) = '' then
      echo    Fcode: ![finfo('!file',-9)].
   else
      echo    Fcode: ![finfo('!file',9)] (![finfo('!file',-9)]).
   endif
   comment ** rec size, eof, flimit **
   echo    Recsize: ![finfo('!file',14)], Eof: ![finfo('!file',19)], &
           Flimit:![finfo('!file',12)].
   comment ** foptions **
   setvar _fopt finfo('!file',-13)
   echo    Foptions: ![finfo('!file',13)] (#!_fopt, ![octal(_fopt)],&
![hex(_fopt)]).
   deletevar _fopt
endif
COMMENT  End of command file.
```

Usage: (assume logged into PUB.SYS)

```
  :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: 103927, Flimit: 108320.
    Foptions: BINARY, FIXED, NOCCTL, STD (#1, %1, $1).

  :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.
  Foptions: ASCII, VARIABLE, CCTL, MSG (#12613, %30505, $3145)

:build temp;temp;rec=40,,f,ascii;disc=100
:file t=temp
: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.
  Foptions: ASCII, FIXED, NOCCTL, STD (#1030, %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.
  Foptions: BINARY, FIXED, NOCCTL, STD (#1050, %2032, $41A).


: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).
```

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*

COBOL Productivity Tricks

Michael S. Vandine
Arizona Industrial Management Systems
2121 W. University Ave. Suite #119
Tempe, Az. 85281

Fourth Generation Languages have received a lot of publicity and acclaim for productivity improvements over regular programming languages, but very little has been written about new tools and "tricks" to improve the productivity of the older programming languages. This paper will address the combination of time-proven techniques such as standardization, development of excellent design documentation and newer techniques unique to the HP3000 including programs from the User Contributed Library that Arizona Industrial Management Systems (AIMS) uses for developing and maintaining systems in COBOL.

The productivity aids that AIMS has developed while producing custom software systems and supporting our timesharing environment are summarized in the following major catagories. As stated above, many of these practices are time-proven and are often taken for granted. Others are contributed library programs, etc. which have been developed by users trying to make their life a little easier.

I. Development of System Standards - Standards must be developed in any data processing endeavor. They are a necessity in order to minimize lost time in any maintenance effort and provide a framework for new development. Everyone involved in program development must be informed of the standards and use them!

   A. The use of copylibs for all files used by a program is absolutely essential. This assures that all programs in the system will reference a data item by the same name. If any changes are made in a field size, a recompile will incorporate the changes in all programs that reference that item.

   B. Naming conventions must be established.

      1. Accounting structure.
      2. Programs.
      3. Stream files.
      4. COBOL source files.
      5. Report files.
      6. Data base sets.

7. View form buffers.
8. Copylib members.

The conventions that you develop are not as important as the fact that naming conventions exist and everyone is USING them!

II. Development of Functional Standards - All systems have many of the same requirements which can be identified, standardized and simplified so that they do not need to be reinvented for each new system. AIMS has identified the following functions for inclusion in all systems which are written.

A. User and Functional Security - Each user is defined to the system for his functional capability and for the terminals to which he has access. Subprograms to maintain users and to validate the logon security are identical for all systems and are copied from a "shell" account to the new system account intact.

B. Report Definitions - Each report to be used in the system is carried on the data base with its default output device, priority, number of copies, security restrictions, etc. When reports are requested, the job stream is generated from the report definition and changes can be made easily to the number of copies, forms messages or routing to an output device. The Report Maintenance and Report Request subprograms are copied from the "shell" account intact.

C. Message Data Base - All messages sent to a user are carried on a Message data base. This allows messages to be changed without a recompile of the source program.

D. Function Key Labels - All function key labels are carried on the data base and a maintenance program exists to allow the user to modify them. These subprograms are never changed from system to system since the screen design standards do not change.

E. Common Routines - Common routines are maintained in the copylib for ease of use in all systems. These include standard centering routines for report headings, standard date conversion routines, etc.

III. Development of a Users Manual - It is an obvious statement that the system must be well defined before it can be programmed, but many people overlook this in the rush to begin writing program code. At AIMS the user manual must be completed and formally approved by the user before any programming is started. A sample of all screens, a definition of each of the data fields, a list of each report with a sample and an explanation of the processing cycle or special considerations are included in the manual.

IV. Development of Preliminary Building Blocks - Once the Users Manual has been completed and approved, the preliminary building blocks of the system need to be assembled. These include:

A. Setting up the accounting structure for the new project.

B. Copying the programs and subprograms from the "shell" account and editing job streams to reflect the new accounting structure.

C. Developing the data base.

D. Completing the screens developed in the documentation process.

E. Using the contributed library program COBGEN to generate the data base and screen buffers for the copylib.

V. Development of Programs - Once the preliminary building blocks have been completed, the programming is started. The online program is a series of subprograms for each screen and/or online function. Subprograms are used to segment the development effort into small, easy to handle modules. Standard add/change/delete masterfile programs exist which are quickly modified to complete master file maintenance functions. All programs are compiled with COBMAP to aid in the use of DEBUG. DEBUG enables a programmer to do more testing per compile.

Each of the above areas is explained in detail in the following pages. When a contributed library program is mentioned, it is explained in detail so a programmer may easily pick up these "tricks" and begin using them immediately.

Developing the System On-line Screens

A. A complete definition of the system is done including a users manual. FORMSPEC is used to design any VPLUS forms that will be used in the online portion of the system. The forms are then written to disc and edited so they are available to the word processing system for inclusion in the users manual. During the initial design stage (completing the users manual), all fields to be used on the form are shown as XXXX or 9999 with no actual field definitions done. This makes the process of system changes during the user approval stage much easier to handle.

   To write a form to disc, a file should be built, a file statement issued before running FORMSPEC, and then the form file (or individual file) must be listed. The format of the build and file equate is:

   ```
   BUILD filename;REC=-82,3,F,ASCII;DISC=10000,32

   FILE FORMLIST=filename,OLD;DEV=DISC;NOCCTL
   ```

B. When the users manual has been approved, all fields shown for the online forms are actually defined in FORMSPEC and all field and finish edits are defined. Meaningful names are used when defining the fields to enable the use of COBGEN as described below. The data base design is finalized and a schema is created using a shell schema file that contains standard data sets for report maintenance, security, etc. and examples of other types of data sets. A couple of "dummy" stand-alone master sets are defined with only one item and only one record in them. This allows them to be used as automatic masters at a future time without data base structuring headaches if system definitions change. Program names are now assigned for all functions required.

Creating the Copylib Members

To create the copylib members for all data base and VPLUS buffers, a user-contributed program named COBGEN is used. When you RUN COBGEN.group.account it will ask for the name of the file you wish to create. Enter a file name, then respond with RETURN's until it asks for a data base name. Type in the data base root file name, then respond with the data base password. For each set on the data base you will be prompted to include or exclude the set. For every set you wish to create a copylib member for, respond with a "Y". Lines will be created with the 01 level of the record being the data set name with a "DB-" in front of it and the 03 levels being each data item name with its corresponding PIC clause. After all sets have been processed, VPLUS buffers may be entered by responding with a form file name when requested. Each form in the form file will be entered into the file. Once all forms have been copied, press the RETURN key until the "PROCEDURE DIVISION" line has been shown. At this point, enter "/EXIT" to exit the program. The file name that you entered initially is now available for you to edit with the editor of your choice.

Delete all of the COBOL lines before the data base buffer definitions and all lines after the VPLUS buffer definitions. Edit the buffers created to conform to your standard naming conventions.

The naming conventions that AIMS has established for data names and the changes made to the generated code are:

A. Data set names - The data set name with a "DB-" in front. No change is needed to these names.

B. Data item names - The data item name from the schema with a prefix indicating the set it is contained in. For example, the VENDOR-NUMBER item in the VENDOR-MASTER set would be defined as VM-VENDOR-NUMBER. This is easily changed by typing "C :03 :,:03 VM-:,line-range".

C. VPLUS buffer names - The screen name or function with a Sxx- in front, with the xx being a sequential number assigned as the copy members are built.

D. VPLUS item names - The Sxx- prefix as described above and the name of the data item that it corresponds to on the data base schema. For example, the VENDOR-NUMBER on a screen would be defined as S01-VENDOR-NUMBER.

E. Since COBGEN doesn't know how a field is to be used, if a field has been defined as a "P" data type, the item will be generated as a COMP-3 field with no decimals. If decimals are needed, these items need to be corrected at this time.

Keep a file for each of the copylib members you wish to create from that portion of the edited file. For example, "K DBVNDMST(30/55)" would keep line numbers 30/55 into a file named DBVNDMST.

The naming conventions AIMS has established for copylib members are:

A. Data base sets - DB plus an abbreviation of the set name.

B. VPLUS buffers - SC plus an abbreviation of the screen name or the screen function.

C. Procedure Division copies - PD plus an abbreviation of the function.

D. Working-Storage copies - WS plus an abbreviation of the function. This abbreviation is the same as the PD abbreviation if this copy is specifically defined for a Procedure Division copy.

Once all files have been created for the copylib members, RUN COBEDIT.PUB.SYS and reference the copylib name where these members will reside. Enter COPY. Respond with one of the file names you have just created. Enter NO to indicate that the file is not in copylib format, then respond with the copylib member name to be created. Perform this procedure until all copylib members have been created.

Now that all of the copylib members are built, a data base has been created and all of the screens are done, it is time to start coding. But wait! Before you go into the EDITOR and type ADD, look for a program that already exists that does somewhat what you need to do. We got tired of "re-inventing the wheel" and set up an account with a basic system that contains a main processing program that takes care of basic log-on security and variable menu processing with calls to subprograms to perform all functions needed in the system. Included in the basic subprograms are a report definition process which allows reports to be defined to the system and a "request report" program which uses the report definition to create a job stream and streams the job for the report. There is also a shell program set up for each major type of program that could be written: add, change, delete of a master data set; add, change, delete of a detail data set; a report program; etc. These programs are written with generic names to be replaced through a global change in the EDITOR. For example, "PROGRAM NAME", "DATA SET TO BE MAINTAINED".

This basic system is moved to the group.account of the new system to be developed. Minor changes are then made to the "shell" copylib to reflect different data base names, etc.

Use of COBOL Macros

Many people have misconceptions about what a COBOL macro really is. It is merely a way of passing parameters to a previously defined collection of COBOL statements. They can be used for virtually any purpose, but are especially helpful for data base and VPLUS calls. Many people have to keep an IMAGE manual at their desk at all times because they have to keep all of the "modes" straight for the opens and calls that they need to do. We have defined all of the opens, gets, opens of terminals, etc. in a file of macros that EVERYONE uses, with descriptive names for each of the macros. For example, %DBGETCHAINFWD is a get of a detail data base set with a mode 5 and %DBGETDIRECT is a get of a data set with a mode 4. These names can, of course, be shortened for even less coding time. The code generated by these macros when a compile is done shows on the compile listing, so you see all moves that are actually being done.

COBOL macros are set up in a file that is accessable to all compile jobs. The format of a macro is very simple. The first line is the definition of the macro name and the actual lines you wish to have copied into your program immediately follow. Each parameter you wish to pass to this code is entered as "!1", "!2", etc. with the numbers corresponding to the positional parameter being passed. The macro set up for a calculated read on a data set looks like this:

```
$DEFINE %DBGC=
       MOVE "!1" TO DB-SET.
       MOVE "!2" TO DB-LIST.
       MOVE "!4" TO DB-KEY.
       MOVE 7 TO DB-MODE.
       CALL "DBGET" USING DATA-BASE, DB-SET,
            DB-MODE, DB-STATUS-ARRAY, DB-LIST,
            !3, DB-KEY.#
```

The format of the actual macro call is as follows:

```
%macro-name(param-1#,param-2#...param-4#)
```

In the example below, lines 010700/010710 are coded by the programmer and lines 048000/048600 are generated from the macro definition, substituting the parameters given in lines 010700/010710.

```
010700     %DBGC(JOB-MASTER#,@#,DB-JOB-MASTER#
010710           ,S09-JOB-NUMBER#)
048000       MOVE "JOB-MASTER" TO DB-SET.
048100       MOVE "@" TO DB-LIST.
048200       MOVE S09-JOB-NUMBER TO DB-KEY.
048300       MOVE 7 TO DB-MODE.
048400       CALL "DBGET" USING DATA-BASE, DB-SET,
048500           DB-MODE, DB-STATUS-ARRAY, DB-LIST,
048600           DB-JOB-MASTER, DB-KEY.
```

By using a standard Working-Storage copylib member whenever data base or VPLUS macros are used, it is much faster to key in only one line and have all of the actual "grunt work" done for you. Also note that to continue a macro on the next line, begin wth the comma right after the pound sign of the previous parameter.

Use of a message data base

All communications sent to the user (displays, VPLUS messages, etc.), are entered into a message data base and all messages are accessed by a message number. This allows the change of a message to be done without re-compiling the program. Also, messages could be translated into a different language and the program would run correctly without recompiling.

Use of Variable Menu Processing

Part of our standard processing for online systems is to have a main program that calls subroutines to show the screens and process the data. The main program uses a GO TO DEPENDING ON statement to determine what subroutine to call. For those unfamiliar with this statement, it transfers control to one of a series of paragraphs depending on the value of a data name. For example, if the following statement is in a program

```
GO TO 1000-HEADER
      2000-DETAIL
      3000-INQUIRE
          DEPENDING ON CHOICE-CODE.
```

and choice-code contains a 2, control is passed to 2000-DETAIL. Therefore, each process performed from a main program may be assigned a process number based on the order that it is defined in the GO TO DEPENDING ON statement.

A "main menu" has been set up which is shown upon entering the main program. The only things contained on the screen are eight boxes at the top containing two lines of eight characters corresponding to each of the function keys. A data set has been set up that contains an entry for the first menu to be shown and any additional menus that may be needed. The items contained in this set consist of actual text describing the processing to be performed when the matching function key is pressed and the process number of the process described. In the example above, the following menu would be set up:

ORDER HEADER	ORDER DETAIL	ORDER INQUIRY	EXIT PROGRAM
01	02	03	99

Each screen in the system has a series of boxes at the top of the screen which are filled by the function labels carried on the data base for that screen name. When a function key is pressed, the GO TO DEPENDING ON statement is performed using the number assigned for that function key. Therefore, from any screen that a user is currently on, there can be a definition allowing him to go to any other process in the system just by pressing a function key. The benefit of this is that the USER may change the screen transfer heirarchy

on any screen just by changing the screen definition on the data base. No recompiling needs to be done and no FORMSPEC changes are required!

Using DEBUG

One of the most powerful tools a programmer has at his disposal is DEBUG. Most programmers do not use this because "It's too hard to understand", "It's too complicated", "I have to be an SPL expert to understand the stack architecture" or "It's easier to just sit down and figure out what's wrong". DEBUG is not difficult or really complicated. You just need to use it a few times to understand what you can get from it. You don't have to use ALL of the functions of DEBUG. By learning a few commands, you will find that you can determine the cause of a problem quickly. You also can, in most cases, find more of the problems at one time by changing input data and bypassing code that is in error. This enables testing to continue without a recompile.

A. Getting a compiled listing from COBMAP.

In order to use DEBUG, you have to have the actual data name and procedure addresses. COBOL allows you to do this by specifying MAP,VERBS on the $CONTROL statement in your program. The only problem with this is that the MAP and VERBS listing is printed AFTER the compile listing. If you need to run a program in DEBUG, it's very difficult to find where you are, where the data names are, etc. because of having to flip back and forth in the listing.

There is a solution! A contributed library program named COBMAP consolidates the MAP and VERB addresses at the end of the program into the actual COBOL listing in the space between the compiler line number and the COBOL line number. Then when you need to run the program in DEBUG, all of the addresses that you need are right with the data names or MOVE statements. Running COBMAP in the compile stream also results in a savings of paper since the MAP and VERB listings aren't printed. A sample job stream using COBMAP follows.

```
!JOB SPS001C,GROUP.ACCOUNT
!PURGE SPS001TL
!BUILD SPS001TL;REC=-132,,F,ASCII;DISC=10000,32
!FILE COBTEXT=SPS001SC,OLD
!FILE COBLIST=SPS001TL;DEV=DISC
!FILE COBUSL=SPS001RB,OLD
!CONTINUE
!RUN COBOLII.PUB.SYS;PARM=7
```

```
!FILE OLDLIST=SPS001TL;DEV=DISC
!FILE NEWLIST=$STDLIST
!FILE NEWTEMP;DISC=10000,32
!CONTINUE
!RUN COBMAP.UTIL.SYS
!PURGE SPS001
!PREP SPS001RB,SPS001;MAXDATA=32000;CAP=IA,BA;PMAP
!IF JCW<WARN THEN
!    SAVE SPS001
!    PURGE SPS001TL
!ENDIF
!EOJ
```

B. Using DEBUG from a VPLUS program.

If the program you need to run in DEBUG happens to be a program that displays screens, you must redirect the screens to another unused terminal. To do this, simply type

"FILE terminal-file-name=device-number"

before running the program. The terminal-file-name may be found in the communication area used for VPLUS applications and is usually "A264X". The device-number is the number of the device you want the screens to appear on. This must be a terminal that is logged off!. You now may enter DEBUG commands at your terminal and do screen responses at the logged off terminal.

C. Basic DEBUG Commands.

Before you use DEBUG you must have a compiled listing with MAP and VERBS in the $CONTROL statement. If the program has more than one section/segment, you will also need a PMAP listing from the PREP to obtain the proper segment number. The following examples will show the basic commands used in DEBUG. These are not all of the commands available, but if you learn these your error-finding capabilities will be greatly increased.

Sample listing from COBMAP:

```
COBOL          EDITOR
 Seq   Address   Seq     COBOL Code

00014  000424  002200 77  INDX          PIC S9(4) COMP.
00015  000460  002300 77  EOC-SW        PIC 9 VALUE ZERO.
   .
   .
   .
00386  001566  013600 1000-MOVE-IN-DATA.
00387          013700
00388  001566  013800      MOVE CD-YEAR-WEEK
00389          013900          TO S05-YEAR-WEEK (INDX).
00390  001607  014000      MOVE CD-AMOUNT
00391          014100          TO S05-AMOUNT (INDX).
00392  001667  014200      ADD 1 TO INDX.
```

To run a program in DEBUG, simply type the run statement and add ";DEBUG" at the end. Once the program loads, the message

DEBUG 0.xxxx

will appear. When this occurs, you are in control and may type any DEBUG command. At this time you need to set "break points"; addresses in the Procedure Division where you want to regain control of the program to look at data, modify data, etc. The format of the "Break" command is:

B segment.offset:@nnn

If there is more than one section in your program, you must look at the PMAP listing to find out what segment number is assigned to the section you want to break in. This may be found right after the paragraph/segment name. If you only have one section, the segment number is zero and it does not have to be specified.

The offset number is the address where you wish to stop. If you wish to stop at the "MOVE CD-YEAR-WEEK" statement, the offset would be 1566.

The ":@" indicates that you want this break to happen every time you reach this address. If this is not specified, the break is only done once, then cleared.

The "nnn" indicates a count. If a number is entered it will do the break after reaching this address "nnn" number of times.

Entering a "B@" will show all breakpoints currently set.

To set a permanent break-point at 1566 type:

B 1566:@

then resume execution of the program by typing:

R

The program will continue executing until this break point has been reached. The following message will be displayed:

BREAK 0.1566

To look at data, the "D" (Display) command is used. command. The format of the "D" command is:

D byte-address/2,number-of-words,display-type

The address for the data name INDX is shown as 424. Since DEBUG uses words, the byte address from the COBOL listing must be divided by 2. The number of words to be displayed will be 1. The display-type is "I" (Integer) for comp fields, "H" (Hex) for comp-3 fields or "A" (ASCII) for display fields. So, to look at the data contained in INDX, type:

D 424/2,1,I

and you will be rewarded with:

DB+212 +0000001

To look at EOC-SW, type:

D 460/2,1,A

and the following would be displayed:

DB+230 0.

Since the field is only one byte long and one word of display was requested, a null-value shows in the

second byte of the display.

Suppose that an error in the processing prevented any further testing. Data items may be changed in DEBUG and the bad code bypassed by changing the current instruction pointer (P register). To change the data item EOC-SW, use an "M" (Modify) command. The format of the "M" command is:

M byte-address/2,number-of-words,type

The address of EOC-SW is 460, the number of words to modify is 1. The type must be specified as "O" (Octal), "I" (Integer) or "A" (ASCII). Modify the EOC-SW data-item by typing:

M 460/2,1,A

DEBUG will show:

230 0.=

Enter a new value by typing:

"1 "

If the new value is surrounded by quotes, it assumes ASCII. If it is preceeded by a pound sign, it assumes Integer. If nothing preceeds the value, Octal is used.

To change the current instruction pointer use the "MR" (Modify Register) command. The format of the "MR" command is:

MR,P

After typing this command, the following will be shown:

P=1667:=

To begin execution at 1000-MOVE-IN-DATA, locate the P register address of that statement on the listing and enter the new address. For the above example, type:

1566

To continue the program without any further breaks,

type:

C@

to clear all break points currently set. An individual break point can also be cleared by typing:

C break-point

An "R" is now done to resume execution of the program.

These basic commands should allow you to quickly determine any logic or data errors that your program encounters and allow you to spend more time coding productively.

D. Running DEBUG from a Dynamic subprogram.

All addresses in the Working-Storage of subprograms declared as "DYNAMIC" are Q relative (see the "$CONTROL" compiler directive in the COBOL manual).

Since all addresses are Q relative, a "DQ" command must be used in place of the "D" command. The Q delta offset for this particular program must be added to the displayed address in order to accomplish the correct offset. This offset can be found at Q-0. If no parameters are passed, Q-0 will be zero.

For example:

```
00023            002300 WORKING-STORAGE SECTION.
00024   000716   002400 77  DEFAULT-CO        PIC 99.
```

To find the Q delta offset, type:

"DQ-0"

Now in order to see what's in 'DEFAULT-CO', type:

DQ716/2+(whatever was in Q-0),1,A

A short way to accomplish this is as follows:

DQ716/2+'Q-0',1,A

CAUTION: Sometimes this doesn't work. You may be one

word off. If things look funny to you, display around a KNOWN data item (something with a "VALUE" clause) and determine the Q delta offset.

All parameters passed to a subprogram are passed to the "Q" area of the stack as addresses pointing to the variables and must be referenced indirectly.

Suppose the following parameters are being passed to the subprogram:

```
PROCEDURE DIVISION USING DATABASE-PARMS
                         VIEW-PARMS
                         SAVE-LPARTN
                         SAVE-MESSAGE
                         DB-SCREEN
                         MISC-PARMS.
```

All parameters start at Q-4 and progress to Q-n where n is the number of parameters passed.

Since they are "pushed" on the Q stack, the last parameter you see in the list is always at Q-4. The one just above it is at Q-5 and so on until you get to the first parmeter in the list. In the above example, "MISC-PARMS" is at Q-4 and "DATABASE-PARMS" is at Q-%11. So, in order to display a variable in any of these parameters, the syntax would be as follows:

DQ-4:,100,A will show the first 100 words of MISC-PARMS.

DQ-11:,100,A will show the first 100 words of DATABASE-PARMS.

If you want to display a specific variable within DATABASE-PARMS, type the following:

DQ-11:+44/2,I -- The colon is used to indicate an indirect address. From this point on, all addresses are exactly the same as you see them in the compiled listing.

DEPARTMENTAL COMPUTING REVISITED

Dennis Vickers
Sunergos
P.O. Box 6914
Orange CA. 92613

The evolution of the data processing function within the organization is a study of change and adaptation. Just when the "right" way to organize is defined and the future of data processing is clear, some new idea or technological breakthrough comes along to muddy the waters. As a result, data processing managers have had to learn to balance a healthy skepticism with a willingness to take a risk. Departmental computing is one of the more popular "new ideas" that data processing managers need to evaluate. This paper is an attempt to more clearly define the idea of departmental computing and see how it may fit into the organization.

The historical background of the data processing function helps to explain its current status. In the mid 1950's new and improved computers became commercially available and new uses for the equipment were being identified. Data processing functions were largely decentralized. Skilled people and resources were placed wherever a need was identified. This was simply the easiest way to handle the new technology, even though it may or may not have been the best economic policy. As hardware became more powerful (and expensive) there came a shift in policy towards centralizing the hardware as well as the staffs assigned to employ it. The processing resources were then divided up as needed among functions within the organization.

To a large extent that policy remains today. Unfortunately this centralization of information processing sometimes had a limiting affect on functional departments. So in the early 1970's the idea of distributed, or departmental, processing began to look good again. This idea was fueled by the introduction of the minicomputer into the business environment. The minicomputer made it economically feasible to physically distribute processing power to functional groups within the organization.

Although minicomputers established a strong position in organizations, the actual implementation of departmental processing never gained widespread acceptance in the 70's. This was primarily due to the lack of good networking technology. The ability to develop corporate-wide information flows was one of the great advantages of centralized data processing and it just was not readily

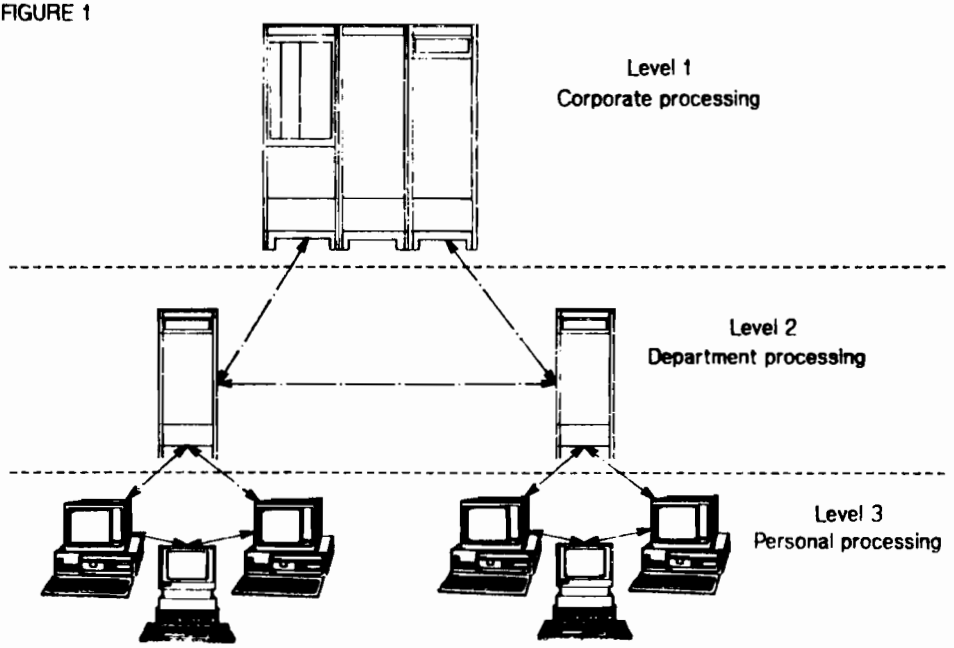
available when processing was decentralized. However, the basic idea behind distributed data processing, putting the processing power and much of the control over the processing into the hands of the people that actually used it, still had many supporters. So departmental processing was put on a back burner waiting for the technology to catch up with the concept.

By the mid 1980's the personal computer (pc) and desktop processing had become firmly entrenched in the organization. The personal computer put unprecedented computing power and flexibility in the hands of individual users. Whether it was part of a formal strategy or not, processing power had been distributed in many organizations. Ironically, the pc's biggest advantage, dedicated personal computing power, was also its biggest disadvantage; it made sharing both data and system resources difficult. The same problem that prevented distributed data processing from taking off, the inability to effectively provide information flows, threatened to limit the pc's usefulness. Fortunately many organizations had taken the lessons learned from earlier attempts at distributing processing power and had developed solutions to many of the networking problems. As a result, hardware and software was becoming commercially available to integrate multiple computers into cohesive computing systems. With these tools in hand an all encompassing strategy for data processing quickly developed. This strategy is commonly referred to as departmental computing.

There is no generally accepted series of rules that precisely defines departmental computing. In fact some people hold that it is more of a philosophy than implementable strategy. The underlying goal is to provide individuals with more powerful technology in order to strengthen the bottom line. This is accomplished by providing individuals with readily available resources. These resources include both processing power and flexible applications. A departmental computing strategy generally involves creating a three-tiered processing network through which data flows back and forth among large centralized mainframes, departmental processors and microcomputers.

In this three-tiered structure, the low end is occupied by the personal computer, the middle tier, by the minicomputer or file server, and the upper level, by the mainframe or host computer. These machines service personal, departmental and corporate-wide processing respectfully (see Figure 1). All levels are extensively connected via high- speed gateways.

FIGURE 1



The lynchpin in this structure is the minicomputer on the departmental level. In a departmental computing strategy these machines are designed to serve as departmental systems for the basic working units in the organization. That is, groups of people that access the same data and perform the same functions on a regular basis. In a survey conducted by Forrestor Research Inc., Cambridge, Mass., it was found that 60% of the information that system users wanted to access is located within the user's department or work group, 25% is somewhere in the user's location, 10% is at a remote location in the company (often a mainframe) and 5% is in a public database like Dow Jones..

In general, systems used on the departmental level have the following major characteristics in common: size, minimal technical requirements, price, peripheral sharing capabilities for PC users, administrative software, and an integrated structure for organizational processing. In terms of size, most entry-level departmental systems have been designed to fit into a unit about the size of a two- drawer file cabinet. These systems blend into the office environment. Of course as processing power grows, so does the size of the system. As far as technical requirements are concerned, these systems are meant to be set up in a normal office with no special environmental requirements. In

addition, the systems can be set up with minimal technical expertise. Routine tasks such as start-up, file backup, and system restart are designed to be fast and simple.

The pricing of departmental systems is also important, they must be priced to accomodate individual departments that often do not have huge capital budgets. The system must offer the ability to share peripherals. The ability to share printers and other such devices is a major attraction. Most departmental systems offer a variety of software for administrative and decision-support functions, electronic mail and messaging, and other office automation products. The most innovative benefit of the departmental system is its ability to easily fit into an integrated structure for organizational processing.

The primary function of these departmental systems is to integrate resources. In fact, George Colony of Forrester Research Inc., Cambridge Mass., refers to these machines as "departmental resource processors" or DRPs. The DRP is defined as an evolved version of the timesharing mini-computer that acts as a pc servant and controls departmental resources such as data bases. It acts as an intermediate between the microcomputer and the host providing a gateway from the department to the mainframe. The DRP also bridges laterally to other departments on a peer-to-peer basis. In attached to the DRP. Instead micro-computers are connected over a high-speed local area network (LAN) to the departmental computer.

The greatest strength of the minicomputer as a departmental system is its ability to play many roles. A few of the functions it could provide to users are distributed data base management, work group administrative support, networking and communications control and pc software distribution. As these new roles are defined it must continue to support current department-wide applications.

There are some definite benefits to implementing a departmental computing structure. Four basic benefits of departmental systems from the user's standpoint have been identified. First, they provide access to a wide variety of corporate information. Second, the departmental computing implementation parallels corporate structures that are based around work groups. The departmental computer also satisfies a user's psychic need for independence. Finally, the systems give middle management a more active role in technological management.

There are also serious problems that need to be addressed

before departmental computing can become a reality. Some of the main areas for concern are security, control, incompatibility, distributed database technology, and organizational resistence. In terms of security, the spreading of sensitive information to distributed databases throughout the organization would make it difficult to protect from unauthorized access. Along this same lines, the reliability and consistency of data comes into question when its control is fragmented throughout the organization.

Even if the security and control issues could be resolved, there is still a question about the type of software required for data base management. A data base management system would need to split the data into three parts: one small portion at the pc, a departmental version for the departmental system, and a complete corporate version on the mainframe. The software to do this is not readily available.

A forth potential stumbling block is the issue of incompatibility. This is a problem that has plagued the computer industry for years. Because micro, mini, and mainframe computers are all part of the departmental computing strategy, it is often necessary to integrate systems from different vendors. Unforturnately pc operating systems like MS/DOS, minicomputer operating systems like MPE and mainframe operating systems like MVS share little compatibility. As a result it is usually not possible to truly integrate the machines. Even when all machines are from the same vendor, compatibility problems often occur.

Finally, the issue of organizational resistance needs to be addressed. Organizational resistance must be looked at from two perspectives. First the personnel considerations. There may be instances of data processing managers who have spent years developing a data processing monopoly based on a powerful mainframe and centralized control. The idea of departmental computing will meet resistance here if it threatens to erode that control. Along the same lines, there may be instances of departmental managers that will resist any changes in their responsibilities. The second concern is that the organization's current information systems may not be well suited for departmental computing. The idea of replacing current systems at considerable cost may outway any benefits gained from implementing the new technology.

While this three-tiered strategy may very well be the wave of the future, no universal ways yet exist to achieve this level of integration. In fact many analysts say that departmental computing will not gain widespread acceptance until the early 1990's when system-level software will catch up to processor

technology. Be that as it may, the time is now for data processing managers to seriously consider what it means to their organization.

Current trends in hardware and software design seem to be leading the way towards departmental computing. Minicomputer vendors are supporting departmental computing by developing systems specifically for that environment. Some recent examples of this include Hewlett-Packard's highly touted Personal Productivity Center, Digital Equipment Corp.'s All in 1 Office and Information System, and Data General's Comprehensive Electronic Office (CEO). These systems are all designed to be solutions to departmental processing needs. They provide software that works in conjunction with microcomputers in the department to integrate departmental resources.

Perhaps the most significant technological advance in this area is the introduction of Intel's 80386 microprocessor. Once these chips get packed into IBM-PC compatible micros and are given the right software, they could act as sophisticated file servers. Intel expects these machines to be priced in the $10,000 to $20,000 range, with performance ratings in the 3 to 4 mips (millions of instructions per second) range. This could provide a price/performance ratio that minicomputer vendors will find hard to match. Combine this with an unobtrusive design that fits naturally into the work environment and the result is an excellent departmental system. The right software is what is currently holding back these systems. But this may be a short-term problem, as Bill Gates, chairman and CEO of Microsoft, says his company is working diligently on an Operating system based on the 80386 chip. Once a widely accepted operating system is in place the applications software will soon follow.

Oddly enough, one of the more popular arguments against implementing a departmental computing strategy has nothing to do with the technology involved. Instead it focuses on how future organizations will most likely be structured. Current trends in organizational theory are leading towards organizations with a flatter pyramidal structure. These futuristic organizations will consist of a number of small business units all reporting to a parent company that functions mainly as a holding company. The argument contends that if this is the case, then the need for a departmental structure will greatly diminish, and along with it, departmental processing. Therefore if formal departments will not exist in future organizations then departmental computing is at best a short term or interim solution.

The argument given above fails in two areas. The first problem is that this argument assumes that a departmental computing strategy is designed to fit only into a departmental structure. In fact this strategy is designed to put processing power in the hands of work groups within organizations. Whether these work groups are project teams, formal departments or autonomous business units, the basic concept still applies.

Secondly, it seems that the central concern of the above argument is that departmental computing will only provide an interim solution. However, in an industry where the technology constantly evolves, and professionals and users in turn become more sophisticated, waiting for a permanent or even long-term solution is frustrating at best. Information systems need to be treated as corporate assets. As such they should be expected to have their own life- cycle. An information systems strategy that has a life- cycle of 7 to 10 years could easily be considered a reasonable investment.

Assuming current trends continue, and the traditionally vertical operations within organizations realign into a series of autonomous, horizontally oriented business units, a new structure for information management will be required. This new structure will coincide with the structure of the organization and centralized, vertical information management will flatten out, becoming more horizontal. The corporate data processing department will have less of a direct role. Instead it will become more of a consultive entity, setting broad strategic goals for the organization. Direct control of information systems will be distributed to data processing groups or knowledgeable individuals on a departmental level.

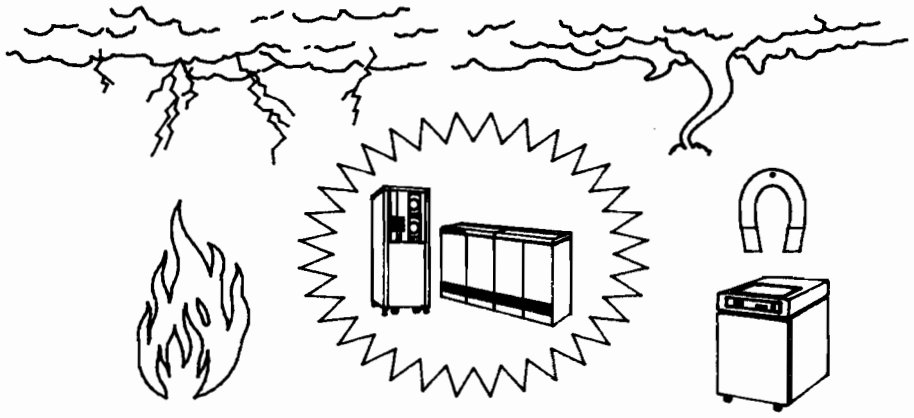
Under this new structure the corporate data processing departments will set standards for a top-down architecture. The architecture will include networking standards, data-base standards, and other broadly defined requirements. The department data processing groups will define bottom-up requirements such as specific applications, local databases, and local area networks. The direction of the bottom-up requirements will generally be defined by the top-down requirements, however some degree of flexibility must be maintained in order to remain responsive to end-user needs.

Responding to end-user needs is the driving force behind departmental computing. Ultimately, only end-user departments can understand how computer and communications systems can be used to gain a competitive edge. In this sense, departmental data processing groups will guide the day-to-day use of the technology inside the corporation. The

corporate data processing department will then become the technologists, ensuring that the strategic technology plan squares with the corporate strategy.

There was once a music professor who held that organ players should not be held in awe for playing music with their hands and feet simultaneously. After all, that's what they get paid to do. The same often holds true for data processing professionals. As departmental computing gains acceptance, data processing managers will be expected to lead their organizations into this still-evolving technology. A resilience to cope with rapid changes will be essential, but then again, that's what data processing managers are paid for.

Insuring the Future of Your Data by Contingency Planning



Leslie Anne Virgilio
Computer Task Group
400 Andrews Street
Rochester, NY 14604

Thomas J. Kaminski
SINGER Education Division
80 Commerce Drive
Rochester, NY 14623

Table of Contents

Section 1. Introduction.

According to Webster's New Students Dictionary, disaster "is an unforeseen, ruinous, and often sudden misfortune that happens either through lack of foresight or through some hostile external agency". To a data processing professional, disaster means the loss of data from a disc-head crash, utility failure, faulty air conditioning, fire, flood, earthquake, hurricane, thunderstorm, tornado, vandalism, sabotage or other occurrences. Most data processing professionals will never see a disaster in their careers and many of them are relying on pure luck to insure the safety of their centers. Any DP center is vulnerable and it will be the unprepared that will panic if disaster strikes. An organized, detailed plan is the key to a successful recovery.

Contingency planning is preparing for disaster (of any kind). A contingency plan should include establishing alternative processing facilities, procedures for restoring critical processing applications on the alternate hardware and establishment of operational procedures, user procedures, and data communications to allow for uninterrupted operation while the original or future site is prepared. A carefully laid contingency plan can significantly improve the ability of a business to survive outages and greatly reduce the length of the outage as well as the cost of recovery.

This paper will not dwell on why a contingency plan is needed. Rather, it will give you enough information to develop your own contingency plan.

Section 2. System Backup Methods.

HP-3000 System Backup

The main method of system backup on the HP-3000 computer is through the MPE commands :SYSDUMP, :FULLBACKUP, and :PARTBACKUP. However, there are other methods including:

Copycat - The HP utility to backup files using removable disc packs

Backpack - A utility from Tymlabs that is a high speed replacement for the MPE SYSDUMP commands

Whichever you use is up to the needs of the individual organization. The application is the same.

There are two types of backups that can be performed:

Complete backup - All files are stored

Relative backup - Only files that have been updated since the last complete backup are stored

The MPE commands :FULLBACKUP and :PARTBACKUP can be used for performing a complete backup and relative backup, respectively. The :SYSDUMP command can be used for both and is sometimes preferred. In the :SYSDUMP command, the system manager can specify a dump list showing what order the files can be put on your backup media. The default for the :FULLBACKUP and :PARTBACKUP commands is:

@.@.@

It may be desirable to have certain file groups on the front of the backup media by using a list like this:

@.@.SYS,@.@.@

This list puts the files in the SYS account at the front of the backup media for quick access in case information must be reloaded.

The schedule you set up for your organization depends on how vital the information on your system is, how often it is updated, and how much time there is to perform system backups. Generally, you should backup up your files on a daily basis. Figure 2-1 shows a typical backup schedule. Note that:

On Monday through Thursday, a relative dump is performed backing up files that have been updated from the last complete backup

On Friday, complete system backups are performed. Since this is the most important backup, two are done. Without a good complete backup, the relative backups following it are useless in most cases. One system backup is stored off site for further protection.

Backups are performed at the end of the user's day to reduce the exposure to a loss of data overnight.

Before performing a backup, excess files should be cleared off the system. This includes system log files and editor work files (K-files). To find these files, enter the following file equations:

```
:LISTF LOG####.PUB.SYS,2
:LISTF K########.@.@,2
```

This will cut down on the amount of space needed to store your files and the amount of time it takes the backup to complete.

All jobs and sessions should be logged off before the start of the backup. During a partial backup, some users can be logged on in certain instances, but they may find that they cannot keep an editor file, run certain programs, or may have very slow response time.

After the backup is complete, check the dump listing. Normally, some files will not be backed up and a message such as this will appear:

```
NOT STORED:  FILE IN USE FOR WRITING
             LOADLIST.PUB.SYS
             LOG____.PUB.SYS
             MEMLOG.PUB.SYS
             SL.PUB.SYS
```

This is normal, these files are being used by the system. Other files may indicate that a session or job was logged on during the backup. If you do not use the standard file list (@.@.@), your statistics will be off at the end (number of files stored, not stored). This is because files that are redundant in the list will be counted as not stored and stored.

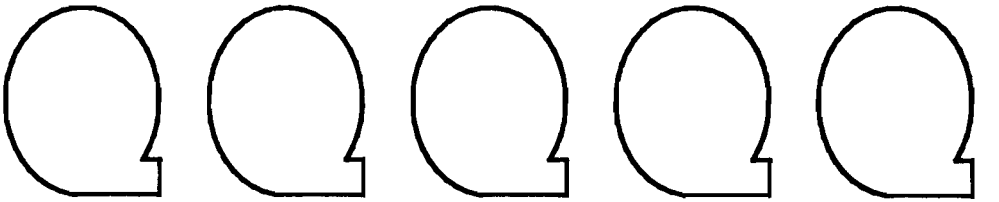
Rotate your backup tapes by using a number of cycles. Four or five cycles are usually good and gives you a month worth of data. Replace your tapes occasionally - ask your tape manufacturer how often.

PC Backup

A primary form of PC backup is using the backup & restore commands to save your data on floppy disk or tape. However, if you have ample room on your HP-3000 disc drives, there is an alternative.

Walker, Richer, and Quinn's series of microcomputer programs (list PC2622) for interfacing to the HP-3000 allow you to transfer groups of files from your microcomputer to the HP-3000. The new reflection software series (Reflection 1, Reflection 3) have a new "plus" option available for using your HP-3000 disc as backup for your microcomputers.

Figure 2-1 Typical Backup Schedule



MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
Partial Backup On-Site Storage	Partial Backup On-Site Storage	Partial Backup On-Site Storage	Partial Backup On-Site Storage	Complete Backup On-site Storage

Complete Backup Offsite Storage

Section 3. Media Storage - On Site, Off Site

Every organization should have good on site and off site storage of materials and information that would be needed in case of a disaster. These include:

Your contingency plan

System backup media (tapes, disc packs, etc.)

Corresponding system backup listings - useful for finding particular files or systems that you would like to selectively load when recovering from a disaster

User and System Documentation - instructions for operation of the computer and software

Software Installation Guides - for any software that would have to be reinstalled on another system

Special Forms - stock paper can be obtained fairly quickly from local vendors, but special forms may take a long time to obtain

A storage rotation plan should be devised and implemented to insure that material in the storage area is rotated on a regular basis. For tapes, it is a good idea to "cycle" your system backups. For example, you can have four sets of tapes - each week, use a different set (1, 2, 3, 4, 1, 2, 3, etc.). This allows you to always have a set of backup tapes in your storage area, even when current backups are being transported and allows you to reload files that you purged some weeks ago and just noticed missing.

Forms should also be rotated. Many forms, especially multi-part forms, can fade or their carbon can dry out making the forms useless. Some vendors will hold back forms for you. This way, the forms are rotated whenever you place an order (the forms held back are shipped and the last set of new forms are held).

On-Site Storage

Storage of backup media and documentation needs to also be done on-site in case something happens to your off-site files and for those emergencies where files need to be reloaded quickly (processing failures, unwanted file purges). Make sure the

on-site storage area that you select is quickly available at all times.

Data Safes are desirable and many sizes are available. Some safes are made for storing paper only. These safes generally keep temperatures below 350 degrees F which is below the flash point of paper, but it is above the melting point of most computer media. A data safe will keep temperatures below 150 degrees F. Safes are rated for number of hours that they will last in a fire.

No matter how good your on-site storage facility, you still must have an off-site facility. A major disaster can wipe out an entire building.

Off-Site Storage

Your off-site storage area should be in another building, away from yours. Some possible locations for an off-site storage facility are:

Another company - you can set up a reciprocal agreement with another company. They will store your off-site media while you store theirs.

Banks - banks are sometimes set up to store backup media for other companies. Safety deposit boxes can be used.

Records Retention Facility - Record Retention Facilities are set up to store both backup media and documentation.

Keep in mind that your backup facility has to be trusted, especially if you are storing crucial data on those files.

Records Retention Facilities can store your media and documentation at a very reasonable cost. For example, prices obtained from a typical facility were as follows:

Storage of Tapes:	.60/tape/month	
Accessions:	.60/item	
Document Storage:	.30/box/month	(12" x 10" x 15")
Accessions:	.60/item	

For example, your full backup takes eight tapes, and you want to store your tapes off-site weekly using four cycles of tapes. You have two boxes with all of your documentation in them and four boxes of special forms. The boxes would be

rotated monthly. Here is what your approximate costs per year would be:

8 tapes x 4 cycles = 32 tapes		
	for 12 months @ .60 =	$ 230.40
16 accessions per week (8 in, 8 out)		
	for 52 weeks @ .60 =	499.20
6 boxes		
	for 12 months @ .30 =	21.60
12 accessions per month (6 in, 6 out)		
	for 12 months @ .60 =	86.40
	Total Yearly Cost	$ 837.60

Check the hours that the retention facility is open. Make sure you can access your information whenever needed.

Section 4. Critical Application Identification

An important part of your Contingency Plan is Critical Application Identification. During a recovery from a disaster, this process serves the following purposes:

Aids in making decisions as to which applications need to be brought up first. This is not a simple priority decision. Some applications are more important than others on certain days and therefore "Peak Processing Periods" must be specified to help with these decisions.

Instructs you on loading the applications onto the computer being used for recovery. Also gives you an idea of what resources the system will take up.

Gives you an indication of special devices needed for certain applications such as printers, tape, special terminals, and other needs. Also, what supplies will be needed.

Allows you to set priorities as to what order systems need to be brought up.

Peak Processing Periods are designated for each system and/or subsystem. This information indicates how long an application can be unavailable before it is needed again. Since this information varies from day to day, it is more or less represented in calendar form. Special processing periods (end of month, quarter, etc.) are also specified. All this is taken into consideration when making judgments about data recovery.

On the next couple of pages are two forms that could be used for Critical Application Identification and Priorities/Peak Processing Periods.

Figure 4-1 Critical Application Identification

Instructions: Enter the name of the system and check if any special terminal types are needed and other equipment such as tapes, printers, plotters, etc. For each task that can be run separately within the system, specify hardware requirements (disk space, terminals, printers, other equipment), restore file sets (0.0.PAYROLL for example), system software needed (i.e. POWERHOUSE, COBOL), and any special forms that may be required including their location and vendor name for reordering.

System: _____

Block Mode [ ] YES	Graphics [ ] YES	Personal [ ] YES
Terminals [ ] NO	Terminals [ ] NO	Computer [ ] NO

Other Equipment: _____

	Disk Space (Sectors)	Number of Terminals	Printer Usage	Other Equip.
Task: _____	_____	_____	_____	_____

Restore File Sets: _____

System Software: _____ File Sets: _____

Special Forms: _____

Location: _____ Vendor: _____

Task: _____	_____	_____	_____	_____

Restore File Sets: _____

System Software: _____ File Sets: _____

Special Forms: _____

Location: _____ Vendor: _____

Task: _____	_____	_____	_____	_____

Restore File Sets: _____

System Software: _____ File Sets: _____

Special Forms: _____

Location: _____ Vendor: _____

Figure 4-2 Priorities/Peak Processing Periods

Instructions: List systems in order of priority. If the computer becomes unavailable on a certain day, at what point must an alternate processing site be obtained? For each system and day, enter the number of hours, name of the day or "NEXT WEEK" that you would need to be recovered by in order for deadlines to be met. Under special processing, list special processing schedules such as end of month, quarter, etc.

System	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Special Processing:							
Special Processing:							
Special Processing:							
Special Processing:							
Special Processing:							
Special Processing:							
Special Processing:							
Special Processing:							
Special Processing:							

Section 5. Recovery Systems.

The following types of disaster recovery systems are available to businesses in the event the existing computer system is no longer accessible.

Private Backup Site

Private backup sites are owned by the business involved. To be of full benefit in the case of a disaster, this site should be in a different location than the original. There are two types of backup sites; "cold" and "hot". A "cold" site is a fully equipped computer facility, without the computer. Only electrical power, air conditioning, and telecommunications equipment exist. When disaster strikes, the computer and required peripherals must be obtained, installed and tested. Although relatively low in cost, the "cold" site has the disadvantage of a lengthy implementation. A "hot" site is a fully equipped computer facility with an identical or very similar computer system to the original, already installed. Obviously, the most desirable system from an operations standpoint, this alternative is extremely expensive. Another drawback to this alternative is the easy justification for using the system/facility for other uses. This eliminates the 100% availability for disaster recovery.

Mutual Backup Agreements

A mutual backup agreement can be between two businesses, or between two different computer sites within the same business, with similar system configurations. They agree to back up one another should a disaster occur. The businesses are usually located near each other. To eliminate competition, the companies are usually in different industries. Although there is little or no cost to the agreement, there are some drawbacks. It is possible, due to the close location of the two sites, that a disaster, such as tornado or earthquake, could occur at both sites. Other problems can arise if one company drastically changes the configuration of their system and the other company does not. Agreements of this type also disturbs the normal processing of the company not affected by the disaster since they will literally have to give their system up to the disastered company for a period of time each day.

Finding potential sites for mutual backup agreements can be done

through your local users group. Set up a strong agreement and make sure that you communicate often with your backup site. When the agreement is with an internal organization, control over the computer environment is often easier.

"Cold" Backup Site

The cold backup site is similar to the privately owned cold backup site. It is an "empty shell" facility owned and operated by a company in the business of data disaster recovery. Unlike the privately owned cold site, this site is available to many companies which could cause competition for its use.

"Warm" Backup Site

Computer service bureaus may offer a warm backup site. Arrangements are usually made in advance to allow the business the use of the service bureau in the event of a disaster. Service bureaus tend to be expensive, and ignore special computer requirements.

"Hot" Backup Site

The hot backup site is often the most acceptable solution to disaster recovery. Similar to the private hot backup site, it is owned and operated by a company in the business of disaster recovery. Although there can be competition for its use, disaster recovery companies can often compensate by having several hot sites strategically located. UPTIME, based in California, has a mobile standalone unit that can be placed wherever needed.

Whatever recovery system you decide on, be sure all your computer needs, printer needs, phone needs, etc. are taken care of. Also, make sure testing of your plan, at the minimum of once a year, can be accommodated.

Section 6. Hardware Replacement.

Hewlett-Packard does not have a written policy for the replacement of hardware. HP sources suggest having a standing purchase order with your local Hewlett-Packard office and in the case of a disaster, they would commit to the shipment of the next available unit. Third party vendors might also be helpful with the availability of used computers and peripherals, as well as new equipment.

Section 7. Disaster Recovery Procedures.

Disaster Recovery Procedures should include all of the tasks that need to be performed when recovering from a disaster and who is responsible for those tasks. Information in this section includes:

Phone Notification List - Use a pyramid chain calling list. An example of this is: Caller A calls 2 people and each of those 2 people call 4 people.

Task Assignment List - A list of tasks that need to be performed in conjunction with disaster recovery. This includes: ordering new hardware and software, insurance notification, and ordering new supplies.

Transportation Plan - Method of transportation of materials and personnel to the recovery site. If transportation includes using automobiles, please be careful not to put any magnetic media near any of the car speaker magnets. When transporting media, try not to expose it to any severe environmental conditions (i.e. cold) as the media will then have to adjust to recovery site conditions before it can then be used.

There are many types of disasters that can occur to a computer center. Here, those types are broken down into three categories:

1. Building Inaccessible - Fire, flood, earthquake, hurricane, tornado, riots, war

2. Computer Area Inaccessible - Vandalism, sabotage, above reasons

3. Computer Inaccessible - processor failure, disk head crash, utility failure, faulty air conditioning

Depending on the category of disaster, you will need to react in a certain way. Here is an example of what a disaster recovery plan might look like:

Severity 1	2	3	Procedure
X	X	X	1. Pyramid Chain Calling - Every contingency plan should have a phone list such as the one described above.
	X	X	2. Have everyone meet together at the office to plan the recovery.
X			3. If the building is inaccessible, meet at the place designated in the systems plan to plan the recovery.
X	X	X	4. Notify your backup processing site at this time if it looks like it will be needed.
	X	X	5. Obtain latest backup media and documentation from on-site storage area.
X	X		6. Obtain latest backup media and documentation from off-site storage area.
X	X	X	7. Hold disaster recovery meeting. Make decisions on which systems to recover first based on information contained in the Contingency Plan and assign tasks as designated.
X	X	X	8. Transport materials using the transportation specified in the contingency plan.
X	X	X	9. Recover systems.

Section 8. Rebuilding Your System on Another Computer.

There are a lot of factors that need to be considered when rebuilding your system on another computer:

Devices - The computer that you are using to recover your data must have the devices required for the systems you need to run.

Operating System - There may be problems if the software you have uses a different operating system than is on the recovery machine (unless you can reload your entire system on an empty machine)

Machine Models - Smaller or busier models of the computer may not be able to handle the volume of data in the same amount of time as your computer did.

System Software - Obtaining Agreements - reference Section 10.

Data Security - You must insure the security of your data. Make sure you clean up all your files after you are done with the recovery system.

Compilers and System Libraries - If your application requires the existence of compilers or SL routines, make sure they are available on your recovery system.

Communication Equipment - Modems, multiplexors, and phone lines are also considerations.

Make sure you know what kind of time is available on the system. With some backup agreements, you may need to set up during odd hours ("C" shift for example).

Make sure you let your backup media adjust to the environment before reloading files on the system. You should probably bring the media into the center as soon as it arrives to facilitate this.

There are two ways to reload files on a system that already has data on it (as in the case of a mutual backup agreement):

Storing off existing files, reloading your files: This takes a lot of time and it is desirable to have fast backup media (such as a removable pack)

Reloading among accounts: To do this, you must have dissimilar account names.

If you are reloading your files on an existing system, there are some helpful programs contained in the Contributed Library:

Account Restructuring Jobs - BULDACCT is a program that builds a stream file that will recreate your account structure on another system. A good idea is to have this run daily. In your plan, document where this file is located (if you put it in PUB.SYS, make sure you put security on it.

UDC Recovery Jobs - There are various programs in the contributed library that will rebuild UDC (User Defined Command) files on your system.

Recovering PC Information

To recover PC information from the HP-3000, make sure your PC transfer program is loaded onto PUB.SYS. With programs such as PC2622, Reflection 1, and Reflection 3, files must be transferred back one at a time. With the "Plus" series from Walker, Richer, and Quinn, you can restore all your files from the disc.

Section 9. Licensed Software Concerns.

Purchased software, whether Hewlett-Packard's or third party, creates another concern in the disaster recovery plan. Licensing agreements prohibit use of the purchased software on any computer other than the one originally purchased for. Disaster has claimed the original computer, now what?

Sources at Hewlett-Packard say that in the case of a disaster, licensed software would be allowed to be used on another system. They warned that software is not compatible between different "MITs" of an operating system. Procedures should be worked out with your local office ahead of time. HP sources did say they would help out in getting the correct version of the licensed software on the recovery system chosen. HP offices keep all versions of licensed software in their local offices.

Sources at COGNOS Corporation said they would be willing to allow movement of their software, including POWERHOUSE products, to a recovery system in the case of disaster. They asked to be contacted before the move is made, if possible. If not possible, they should be contacted the next working day. COGNOS' software is dependent on the series of HP-3000. If you plan to use a different series as a recovery system, prior arrangements should be made with COGNOS.

The assumption might be made that other software vendors have similar policies. The safest thing to do would be to contact any software vendors you deal with when you are developing your disaster recovery plan.

Section 10. Supplies and Auxiliary Equipment.

Also important is all of the computer supplies and other equipment needed to run your systems. If your supplies and equipment have been destroyed, you need to order these items. Your Contingency Plan should contain a list of vendors, purchase order numbers, inventory lists, and other information that would facilitate such a replacement.

A suggested vendor identification form can be found in Figure 10-1. This form shows the name, address, and phone number of the vendor; associated purchase order or account numbers; and items provided by that vendor including standard quantities, prices, and delivery times.

Figure 10-1 Vendor Identification Form

Vendor: _____ PO#/Acct# _____

Address: _____ Descript _____

_____ PO#/Acct# _____

_____ Descript _____

Contact: _____ Phone: _____

Part No	Description	Standard Quantity	Cost	Delivery Time

Section 11. Concluding Thoughts.

Contingency plans should be well though out. Thoroughness in testing will save much time and reduce panic in the event of a disaster. Periodic updating of your plan will preserve its integrity.

Remember, a contingency plan is your only key to insuring the future of your data.

HOW PROGRAMMING LANGUAGES DIFFER:

A CASE STUDY OF SPL, PASCAL, AND C

Eugene Volokh
VESOFT, Inc.
1135 S. Beverly Dr.
Los Angeles, CA 90035 USA
(213) 282-0420

INTRODUCTION

Programmers get passionate about programming languages. We spend most of our time hacking code, exploiting the language's features, being bitten by its silly restrictions. There are dozens of languages, and each one has its fanatical adherents and its ardent detractors. Some like APL, some like FORTH, LISP, C, PASCAL; some might even like COBOL or FORTRAN, perish the thought.

In particular, a lot of fuss has recently arisen about SPL, PASCAL, and C. All three of them are considered good "system programming" (whatever that is) languages, and naturally people argue about which one is the best.

HP's Spectrum project has come out in favor of PASCAL -- all new MPE/XL code will be written in PASCAL, and HP won't even provide a native mode SPL compiler. On the other hand, HP's also getting more and more into UNIX, which is coded entirely in C. Especially between C and PASCAL adherents there seems to be something like a "holy war"; it becomes not just a matter of advantages and disadvantages, but of Good and Evil, Right and Wrong. Strict type checking is Good, some say -- loose type checking is Evil; pointers are Wrong -- array indexing is Right. The battle-lines are drawn and the knights are sharpening their swords.

But, some ask -- what's the big deal? After all, it's an axiom of computer science that all you need is an IF and a GOTO, and you can code anything you like. Theoretically speaking, C, SPL, and PASCAL are all equivalent; practically, is there that much of a difference?

In other words, is it just esthetics or prejudice that animate the ardent fans of C, PASCAL, or SPL, or are there real, substantive differences between the languages -- cases in which using one language rather than another will make your life substantially easier? Are the main differences between, say, C and PASCAL that PASCAL uses BEGIN and END and C uses "{" and "}"? That C's assignment operator is "=" and PASCAL's is ":="?

The goal of this paper is to answer just this question. I will try to analyze each of the main areas where SPL, C, and PASCAL differ, and point out those differences using actual programming examples. I'll try not to emphasize vague, general statements, like "PASCAL does strict type checking", or subjective opinions, like "C is too hard to read"; rather, I want to use SPECIFIC EXAMPLES which can help make clear the exact influence of strict or loose type checking on your programming tasks.

Saying that I'll "compare SPL, PASCAL, and C" isn't really saying a whole lot. How will I compare them? What criteria will I use to compare them? Will I compare how easy it is to read them or write them? Will I compare what programming habits they instill in their users? Which versions of these languages will I compare?

To do this, and to do this in as useful a fashion as possible, I set myself some rules:

* I resolved to try to show the differences by use of examples, preferrably as real-life as possible. The emphasis here is on CONCRETE SPECIFICS, not on general statements such as "C is less readable" or "PASCAL is more restrictive".

* I decided not to go into questions of efficiency. Compiling a certain construct using one implementation of a compiler may generate fast code, whereas a different implementation may generate slow code. Sure, the FOR loop in PASCAL/3000 may be less efficient than in SPL or in CCS's C/3000, but who knows how fast it'll be under PASCAL/XL?

  For this reason, I don't wax too poetic about the efficiency advantages of features such as C's "X++" (which increments X by 1) -- a modern optimizing compiler is quite likely to generate equally fast code for "X:=X+1", automatically seeing that it's a simple increment-by-one (even the 15-year-old SPL/3000 compiler does this).

  The only times when I'll mention efficiency is when some feature is INHERENTLY more or less efficient than another (at least on a conventional machine architecture); for instance, passing a large array BY VALUE will almost certainly be slower than passing it BY REFERENCE, since by-value passing would require copying all the array data.

  Even in these cases, I try to play down performance considerations; if you're concerned about speed (as well you should be), do your own performance measurements for the features and compiler implementations that you know you care about.

* I resolved -- for space reasons if for no other -- not to be a textbook for SPL, PASCAL, or C. Some of the things I say apply equally well to almost all programming languages, and I hope that they will be understandable even to people who've never seen SPL, PASCAL, or C.

  For other things, I rely on the relative readability of the languages and their similarity to one another. I hope that if you know any one of SPL, PASCAL, or C, you should be able to understand the examples written in the other languages.

  However, it may be wise for you to have manuals for these three languages -- either their HP 3000 implementations or general standards -- at hand, in case some of the examples should prove too arcane.

* As you can tell by the size of this paper, I also decided to be as thorough as practical in my comparisons, and ESPECIALLY in the evidence backing up my comparisons.

  One of the main reasons I wrote this paper is that I hadn't seen much OBJECTIVE discussion comparing C and PASCAL; I wanted not just to present my conclusions -- which might as easily be based on prejudice as on fact -- but also the reasons why I arrived at them, so that you could decide for yourself.

  So as not to burden you with having to read all 200-odd pages, though, I've summarized my conclusions in the "SUMMARY" chapter. You might want to have a look there first, and then perhaps go back to the main body of the paper to see the supporting evidence of the points I made.

WHAT ARE C AND PASCAL, ANYWAY?

If you think about it, SPL is a very unusual language indeed. To the best of my knowledge, there is exactly one SPL compiler available anywhere, on any computer (eventually, the independent SPLash! may be available on Spectrum, but that is another story). I can say "SPL supports this" or "SPL can't do that" and, excepting differences between one chronological version of SPL and the next, be absolutely precise and objectively verifiable. SPL can be said to "support" something only because there is only one SPL compiler that we're talking about.

To say "PASCAL can do X" is a chancy proposition indeed. ANSI Standard PASCAL doesn't support variable-length strings, but most modern PASCAL implementations, including HP PASCAL, have some sort of string mechanism. What about HP's new PASCAL/XL, reputed to be even more powerful still? Similarly, with C, there are the old "Kernighan & Ritchie" C, the proposed new ANSI standard C, whatever it is that HP uses on the Spectrum, AND whatever you use on the 3000, which might be CCS's C compiler or Tymlabs' C.

On the one hand, I contemplated comparing standard C and standard PASCAL. This is easier for me, and it also makes sense from a portability point of view (if you want it to be portable, you're better off using the standard, anyway).

On the other hand, portability is fine and dandy, but most people aren't going to be porting their software any further than from an MPE/XL machine to an MPE/V machine and back. As long as you stick to HP 3000s, you have the full power of so-called "HP PASCAL", an extended superset of PASCAL that's supported on 3000s, 1000s, 9000s, and the rest; it's hardly fair (or practical) to ignore this in a comparison.

Finally, what about PASCAL/XL? It'll have even more useful features, but they may not be ported back to the MPE/V machines, at least for a while. Should I then compare PASCAL/XL and C/XL, a representative contest for the XL machines, but not necessarily for MPE V machines, and certainly not if you really want to port your software onto other machines.

This is all, incidentally, aggravated by the fact that HP's extensions to PASCAL are more substantial than its extensions to C; thus, comparing the "standards" is likely to put PASCAL in a relatively worse light than comparing "supersets" (not to say that PASCAL is worse than C in either case).

Faced with all this, I've decided to compare everything with everything else. There are actually 7 different compilers I discuss at one time or another:

* SPL.
  There's only one, thank God.

* Standard PASCAL.
  This is the original ANSI Standard, on which all other PASCALs are based. This is also very similar to Level 0 ISO Standard PASCAL (see next item).

* Level 1 ISO Standard PASCAL.
  This standard, put out in the early 1980's, supports so-called CONFORMANT ARRAY parameters (see the DATA STRUCTURES chapter). The same standard document defined "Level 0 ISO Standard PASCAL" to be much like classic

"Standard PASCAL", i.e. without conformant arrays. Compiler writers were given the choice of which one to implement, and it isn't obvious how popular Level 1 ISO Standard will be. When I say "Standard PASCAL", I mean the original standard, which is almost identical to the ISO Level 0 Standard.

* PASCAL/3000.
  This is HP's implementation of PASCAL on the pre-Spectrum HP 3000. Although the Spectrum machines will also be called 3000's, when I say PASCAL/3000 I mean the pre-Spectrum version. PASCAL/3000 is itself a superset of HP Pascal, which is also implemented by HP on HP 1000s and HP 9000s. PASCAL/3000 is a superset of the original Standard PASCAL, not the ISO Level 1 Standard.

* PASCAL/XL.
  This is HP's implementation of PASCAL on the Spectrum. It's essentially a superset of both PASCAL/3000 and the ISO Level 1 Standard.

* Kernighan & Ritchie (K&R) C.
  This is the C described by Brian Kernighan and Dennis Ritchie in their now-classic book "The C Programming Language" (which, in fact, is usually called "Kernighan and Ritchie"). Although never an official standard, it is quite representative of most modern C's. In fact, for practical purposes, it can be said that a program written in K & R C is portable to virtually any C implementation (assuming you avoid those things that K&R itself describes as implementation-dependent).

* Draft ANSI Standard C.
  ANSI is now working on codifying a standard of C, which will have some (but not very many) improvements over K&R. My reference for this was Harbison & Steele's book "C: A Reference Manual", which also discusses various other implementations of C. Although Draft ANSI Standard C is Standard, it is also Draft. Some of the features described in it are implemented virtually nowhere, and it's not clear how much of them C/XL will include.

Matters are further complicated, of course, by the lack of an HP-provided C compiler on the pre-Spectrum HP 3000. The compiler I used to research this paper is CCS Inc.'s C/3000 compiler, which is a super-set of K&R C and a subset of Draft ANSI Standard C. The most conspicuous Draft Standard feature that CCS C/3000 lacks is Function Prototypes -- an understandable lack since virtually all other C compilers don't have them, either.

Whenever any difference exists between any of the PASCAL or C versions, I try to point it out. Which versions you compare are up to you:

* You can compare Standard PASCAL and K&R C.
  If it isn't in these general standards that everybody implements, you're unlikely to get much portability.

* You can compare PASCAL/XL and Draft ANSI Standard C.
  These are the compilers that will most likely be available on the Spectrum.

* You can compare PASCAL/3000 and Draft ANSI Standard or K&R C
  Even though you might not usually care about porting to, say, an IBM or a VAX, you may very seriously care about porting from the pre-Spectrum to the Spectrum and vice versa. HP hasn't promised to port PASCAL/XL back to the pre-Spectrums, so PASCAL/3000 is probably the lowest common denominator.

SPL is nice. At least until SPLash!'s promised Native Mode SPL compiler comes out, there's only one SPL compiler to compare with. This makes me very happy.

ARE C, PASCAL, AND SPL FUNDAMENTALLY DIFFERENT OR FUNDAMENTALLY ALIKE?

In my opinion, they are definitely FUNDAMENTALLY ALIKE. In the rest of the paper, I'll tell you all about their differences, but those are EXCEPTIONS in their fundamental similarity.

Why do I think so? Well, virtually every important construct in either of the three languages has an almost exact parallel in the other two (the only exception being, perhaps, record structures, which SPL doesn't have).

* All three languages emphasize writing your program as a set of re-usable, parametrized procedures or functions (which, for instance, COBOL 74 and most BASICs do not);

* All three languages share virtually the same rich set of control structures (which neither FORTRAN/IV nor BASIC/3000 posess).

* The languages may on the surface LOOK somewhat different (PASCAL and C certainly do), but remember that the ESSENCE is virtually identical -- PASCAL may say "BEGIN"

and "END" where C says "{" and "}", but that's hardly a SUBSTANTIVE difference.

Despite all the differences which I'll spend all these pages describing -- and I think many of the differences are indeed very important ones -- I still think that SPL, PASCAL, and C are about as close to each other as languages get.

SO, WHICH IS BETTER -- C, PASCAL, OR SPL?

You think I'm going to answer that? With all my pretensions to objectivity, and dozens of angry language fanatics ready to berate me for choosing the "wrong one"?

The main purpose of this paper is to show you all the differences and let you decide for yourselves; after all, there are so many parameters (how portable do you want the code to be? how much do you care about error checking?) that are involved in this sort of decision.

The closest I come to actually saying which is better is in the "SUMMARY" chapter (at the very end of the paper); there I explain what I think the major drawbacks and advantages of each language are. Look there, but remember -- only you can decide which language is best for your purposes.

HOW TO GET THE FULL TEXT OF THIS PAPER

In my researches, I tried to be as thorough as possible. In fact, I discussed in some detail:

* CONTROL STRUCTURES in all three languages.
* DATA STRUCTURES.
* I/O.
* STRINGS.
* SEPARATE COMPILATION.
* BUILT-IN LANGUAGE OPERATORS.
* PROCEDURES WITH VARIABLE NUMBERS OF PARAMETERS.
* VARIABLES THAT POINT TO PROCEDURES (!).

* MACRO FACILITIES -- SPL DEFINE'S AND C #DEFINE'S.

* POINTERS.

* "LOW-LEVEL" MACHINE ACCESS FACILITIES -- DEALING WITH REGISTERS, ASSEMBLY, ETC.

* THE C STANDARD I/O LIBRARY (a powerful toolbox).

Needless to say, this made for a very large paper, too large to print in its entirety in these Proceedings. It is available from VESOFT at:

VESOFT, Inc.
1135 S. Beverly Dr.
Los Angeles, CA 90035, USA
(213) 282-0420

However, as you see below, the summary has also been reproduced here.

SUMMARY

If you have not yet guessed, I am by nature a loquacious man. For every issue I've raised, I've spent pages providing examples, giving arguments, discussing various points of view.

This was all intentional; rather than just presenting my own opinions, I wanted to give as many of the facts as possible and let you come to your own conclusions. However, this resulted in a paper that was 200-odd pages long -- not, I would conjecture, the most exciting and tittilating 200 pages that were ever written.

In this section I want to present a summary of what I think the various merits and demerits of SPL, PASCAL, and C are. All of the things I mention are discussed in more detail elsewhere in the paper, so if you want clarification or evidence, you'll be able to find it. I hope, though, that these lists themselves might put all the various arguments and alternatives in better perspective.

Remember, however, as you read this -- if this all sounds opinionated and subjective, all the evidence is elsewhere in the paper, if you want to read it!

THE TROUBLE WITH SPL

[This section includes all those things that make SPL hard to work in. This isn't just "features that exist in other languages but not in SPL" -- these are what might be considered drawbacks (serious or not), things that you're likely to run into and regret while programming in SPL.]

* SPL IS COMPLETELY NON-PORTABLE. There is no HP-supplied Native Mode SPL on Spectrum, and certainly not on any other machine. (Note: Software Research Northwest intends to have a Native Mode SPL compiler released by MAY 1987.)

* SPL'S I/O FACILITY FRANKLY STINKS. Outputting and inputting either strings or numbers is a very difficult proposition -- I think this is the major reason why more HP 3000 programmers haven't learned SPL.

* SPL HAS NO RECORD STRUCTURES. This is a severe problem, but not fatal -- there are workarounds, though none of them is very clean. See the "DATA STRUCTURES" section for more details.

THE TROUBLE WITH STANDARD PASCAL

* STANDARD PASCAL'S PROCEDURE PARAMETER TYPE CHECKING IS MURDEROUS:

  - YOU CAN'T WRITE A GENERIC STRING PACKAGE OR A GENERIC MATRIX-HANDLING PACKAGE because the same procedure can't take parameters of varying sizes! That's right -- you either have to have all your strings be 256-byte arrays (or some such fixed size), or have a different procedure for each size of string! Try writing a general matrix multiplication routine; it's even more fun.

  - YOU CAN'T WRITE A GENERIC ROUTINE THAT HANDLES DIFFERENT TYPES OF RECORD STRUCTURES OR ARRAYS FOR ARGUMENTS. Say you want to write a procedure that, say, does a DBPUT and aborts nicely if you get an error; or does a DBGET; or does anything that might cause it to want to take a parameter that's "AN ARRAY OR A RECORD OF ANY TYPE". You can't do it! You must have a different procedure for each type!

  - YOU CAN'T WRITE A WRITELN-LIKE PROCEDURE THAT TAKES INTEGERS, STRINGS, OR FLOATS (perhaps to format them all in some interesting way).

* IN STANDARD PASCAL, YOUR PROGRAM AND ALL THE PROCEDURES IT CALLS MUST BE IN THE SAME FILE! That's right -- if your program is 20,000 lines, it must all be in one file, and all of it must be compiled together.

* STANDARD PASCAL HAS NO BUILT-IN STRING HANDLING FACILITIES, AND NO MECHANISM FOR YOU TO IMPLEMENT THEM. Not only are simple things like string comparison, copying, etc. (which are built into SPL) missing; you can't write generic string handling routines of your own unless all your strings have the same length and occupy the same amount of space (see above)!

* STANDARD PASCAL'S I/O FACILITY IS ABYSMAL.

- YOU CAN'T WRITE A STRING WITHOUT CAUSING THE OUTPUT TO DEVICE TO GO TO A NEW LINE (i.e. you can't just "prompt the user for input" and have the cursor remain on the same line).

- YOU CAN'T OPEN A FILE FOR APPEND OR INPUT/OUTPUT ACCESS.

- YOU CAN'T OPEN A FILE WITH A GIVEN NAME. So you want to prompt the user for a filename and open that file? Tough cookies -- Standard PASCAL has no way of letting you do that.

- IF YOU PROMPT THE USER FOR NUMERIC INPUT, THERE'S NO WAY FOR YOUR PROGRAM TO CHECK IF HE TYPED CORRECT DATA. Say you ask him for a number and he types "FOO"; what happens? The program aborts! It doesn't return an error condition to let you print an error and recover gracefully -- it juts aborts!

- SIMILARLY, IF YOU TRY TO OPEN A FILE AND IT DOESN'T EXIST (or some such file system error occurs on any file system operation), YOU DON'T GET AN ERROR CODE BACK -- YOU GET ABORTED! What a loss! - YOU CAN'T DO "DIRECT ACCESS" -- READ OR WRITE A PARTICULAR RECORD GIVEN ITS RECORD NUMBER. Think about it -- how can you write any kind of disc-based data structure (like a KSAM- or IMAGE-like file) without some direct access facility? Even FORTRAN IV has it!

* OTHER, LESS PAINFUL, BUT STILL SIGNIFICANT LIMITATIONS INCLUDE: (These are things which you can certainly live without, unlike some of the problems above, which can be extremely grave. However, although you can live without them, they are still desirable, and in Standard PASCAL -- partly because of its restrictive type checking -- you CAN'T emulate them with any degree of ease. Their lack, incidentally, is felt particularly in writing large system programming applications.)

- STANDARD PASCAL DOESN'T ALLOW YOU TO DYNAMICALLY ALLOCATE A STRING OF A GIVEN SIZE (where the size is not known at compile-time). PASCAL talks much about its NEW and DISPOSE functions, which dynamically allocate space at run-time.

  These functions are certainly very useful, and are in fact essential to many systems programming applications. However, say you want to allocate an array of X elements, where X is not known at compile-time -- YOU CAN'T! You can allocate an array of, say, 1024 elements, forbidding X to be greater

than 1024 and wasting space if X is less than 1024; you CAN'T simply say "give me X bytes (or words) of memory".

- STANDARD PASCAL HAS NO REASONABLE MECHANISM FOR DIRECTLY EXITING OUT OF SEVERAL LAYERS OF PROCEDURE CALLS. Say that your lowest-level parsing routine detects a syntax error in the user's input and wants to return control directly to the command input loop (the larger and more complicated your application, the more common it is that you want to do something like this).

  "Un-structured" as this may seem, it can be quite essential (see the "CONTROL STRUCTURES -- LONG JUMPS" chapter), and Standard PASCAL provides only very shabby facilities of doing this.

- STANDARD PASCAL HAS NO WAY FOR HAVING VARIABLES THAT POINT TO FUNCTIONS AND PROCEDURES. Strange as this may seem, variables that point to procedures/ functions can be VERY useful -- see the chapter on "PROCEDURE AND FUNCTION VARIABLES" for full details. Interestingly, even Standard PASCAL recognizes their utility by implementing PARAMETERS that point to procedures and functions, but it doesn't go all the way and let arbitrary VARIABLES do it.

If you respond that many PASCALs fix many of these drawbacks, I'll agree -- BUT WHAT HAPPENS TO PORTABILITY? If you use PASCAL/3000's string handling package (a pretty nice one, too), how are you going to port your program to, say, a PC implementation that has a different string handling package? What's more, some implementations -- like PASCAL/3000 itself -- don't solve many of the most important problems listed above!

THE TROUBLE WITH KERNIGHAN & RITCHIE C

* WHERE STANDARD PASCAL'S PROCEDURE PARAMETER TYPE CHECKING IS TOO RESTRICTIVE, K&R C'S IS NON-EXISTENT! If you write a procedure

```
p (x1, x2, x3)
int x1;
char *x2;
int *x3;
```

and call it by saying

```
p (13.0, i+j, 77, "foo")
```

the compiler won't utter a peep. Not only won't it automatically convert 13.0 to an integer -- it won't print an error message about that, OR that "I+J" is not a character array, OR that 77 is not an integer pointer (which probably means that P expects X3 to be a by-reference parameter and you passed a by-value parameter), OR EVEN THAT YOU PASSED THE WRONG NUMBER OF PARAMETERS!

* WHILE NOT AS BAD AS PASCAL'S, K&R C'S I/O FACILITIES STILL HAVE SOME MAJOR LACKS. Most serious are:

  - NO DIRECT ACCESS (read record #X).

  - NO INPUT/OUTPUT ACCESS.

* K&R C, THOUGH FAIRLY STANDARD, HAS NO STANDARD STRING PACKAGE. Unlike in Standard PASCAL, though, it's fairly easy to write, since you CAN write a C procedure that takes, say, a string of arbitrary length.

* C IS UGLY AS SIN. At least that's what some PASCAL programmers say; C programmers obviously disagree. People complain that C is just plain UGLY and thus (subjectively) difficult to read; they talk about everything from the "{" and "}" that C uses instead of "BEGIN" and "END" to C's somewhat arcane operators, like "+=" and "--". I'm not saying that this is either TRUE or FALSE; unfortunately, it's much too subjective to discuss in this paper.

  However, don't be surprised if you decide that on all the merits, C is superior but you can't stand writing with all these funny special characters; or, that PASCAL is the best, but it's much too verbose for you!

IS ISO LEVEL 1 STANDARD PASCAL ANY BETTER THAN THE ANSI STANDARD?

* THE ONLY DIFFERENCE BETWEEN ISO LEVEL 1 STANDARD PASCAL AND THE ANSI STANDARD (what I call simply "Standard PASCAL") IS THAT IT ALLOWS YOU TO WRITE PROCEDURES THAT TAKE ARRAYS OF VARIABLE SIZES. This eliminates one of the worst problems in Standard PASCAL -- that you can't write a generic string handling package, or a matrix multiplication routine, etc.; however, all the other

problems (lack of separate compilation, bad I/O, etc.) still remain.

* NOTE THAT IT'S NOT CLEAR HOW MANY NEW PASCAL COMPILERS WILL FOLLOW THE ISO LEVEL 1 STANDARD. The ISO Standard document makes it clear that implementing this feature is optional (without them, a compiler will conform only to the "ISO Level 0 Standard"); PASCAL/3000 doesn't implement it, but PASCAL/XL does.

IS PASCAL/3000 ANY BETTER THAN ANSI STANDARD PASCAL?

* PASCAL/3000 supports:

  - A PRETTY GOOD STRING PACKAGE.

  - IMPROVED, though still somewhat difficult, I/O.

  - THE ABILITY TO COMPILE A PROGRAM IN SEVERAL PIECES.

  - THE ABILITY TO WRITE A PROCEDURE OR FUNCTION THAT TAKES A STRING (BUT NOT ANY OTHER KIND OF ARRAY) OF VARIABLE SIZE.

* The remaining problems still include:

  - PARAMETER TYPE CHECKING STILL WAY TOO TIGHT. You still can't write a procedure that takes an integer array of arbitrary size or an arbitrary array/record (say, to do DBGETs or DBPUTs with); you still can't write, say, a matrix multiplication routine (just as an example).

  - I/O STILL HAS PROBLEMS:

    * IT'S STILL VERY DIFFICULT (not impossible, but still very painful) TO TRAP ERROR CONDITIONS, SUCH AS FILE SYSTEM ERRORS OR INCORRECT NUMERIC INPUT.

    * YOU CAN'T OPEN A FILE USING "FOPEN" AND THEN USE THE PASCAL I/O SYSTEM WITH IT. This means that any time you need to specify a feature that PASCAL's OPEN doesn't have (such as "open temporary file", "build a new file with given parameters", "open a file on the line printer", etc.), you can't just call FOPEN and then use PASCAL's I/O facilities. You either have to do all FOPEN/FWRITE/FCLOSEs, or you have to issue a :FILE equation, which is difficult and still doesn't give you all the features you want.

- THE "LESS IMPORTANT BUT STILL SUBSTANTIAL" LIMITATIONS STILL EXIST -- it's hard to allocate variable-size strings, you can't immediately exit several levels of nesting, and you can't have variables that point to functions or procedures.

* REMEMBER -- YOU CAN'T COUNT ON "A PARTICULAR IMPLEMENTATION" TO SAVE YOU HERE! If you could live with Standard PASCAL's restrictions by knowing that, say, string handling or a good I/O facility would surely be implemented by any particular implementation, remember: PASCAL/3000 is a particular implementation! If you run into a restriction with PASCAL/3000, that's it; you either have to work around it or use a different language.

IS PASCAL/XL ANY BETTER THAN THE ANSI STANDARD?

Surprisingly, yes. ALL OF THE MAJOR PROBLEMS I POINTED OUT IN STANDARD PASCAL SEEM TO HAVE BEEN FIXED IN PASCAL/XL! The only words of caution are:

* IT MAY BE GREAT, BUT IT'S NOT PORTABLE -- NOT EVEN TO PRE-SPECTRUMS! HP still hasn't announced when (if ever) it'll implement all of PASCAL/XL's great features on the pre-Spectrum machines. As long as it doesn't, you'll have to either avoid using of all PASCAL/XL's wonderful improvements, or be stuck with code that won't run on pre-Spectrum 3000's!

* BE SKEPTICAL. "New implementations" always look great, precisely because we haven't had the chance to really use them. For all we know, the compiler may be ridden with bugs, or it might be excruciatingly slow in compiling your program, or it might generate awfully slow code! Even more likely, there may be serious design flaws that make programming difficult -- it's just that we won't notice them until we've programmed in it for several months! As I said, BE SKEPTICAL.

IS DRAFT ANSI STANDARD C BETTER THAN KERNIGHAN & RITCHIE C?

Again, it seems it might be! It's standardized the I/O and string handling facilities (and they're pretty good ones at that), AND it's implemented some nice-looking parameter

checking. Still, beware:

* BEING A "DRAFT STANDARD", IT MIGHT BE YEARS (OR DECADES) BEFORE ALL OR MOST C COMPILERS HAVE ALL OF ITS FEATURES. Note, however, that most modern C compilers already include some of the Draft Standard's new features, except for the strengthened parameter checking, which is still relatively rare.

* IF YOU THOUGHT KERNIGHAN & RITCHIE C WAS UGLY, YOU'LL STILL THINK THIS ABOUT DRAFT STANDARD C. I don't want to imply that K&R C IS ugly -- it's just that many old SPL, PASCAL, and ALGOL programmers think so. It may not be objectively demonstrable, or even objectively discussible; however, that's the reaction I've seen in some (more than a few!) people. All I can say is this -- if you suffer from it, the Draft Standard still won't help you.

NICE FEATURES THAT SOME LANGUAGES DON'T HAVE AND OTHERS DO

The "PROBLEMS WITH" sections discussed things that could make programming in SPL, PASCAL, or C a miserable experience. It emphasized some things that were show-stoppers and others that simply frayed on the nerves; one thing it conspicuously EXCLUDED were the good features that you could live without, but would rather have. The following is a summary of all these, plus some of the things we've already mentioned above.

[Legend: "STD PAS" = Standard PASCAL or ISO Level 1 Standard;
"STD C" = Draft Proposed ANSI Standard;
"YES" = good implementation of this feature;
"YES+" = excellent or particularly nice implementation;
"YES-" = OK, so they've got it, but it's rather ugly;
"NO" = no;
"HNO" = Hell, no!;
"---" = Major loss! No support of REALLY IMPORTANT feature]

	STD PAS	PAS/ 3000	PAS/ XL	K&R C	STD C	SPL
RECORD STRUCTURES	YES	YES	YES	YES	YES	NO
STRINGS	---	YES+	YES+	YES-	YES+	YES

ENUMERATED DATA TYPES (see "DATA STRUCTURES")	YES	YES	YES	NO	YES-	NO
SUBRANGE TYPES (see "DATA STRUCTURES"; may not be all that useful)	YES	YES	YES	NO	NO	NO
OPTIONAL PARAMETER/VARIABLE NUMBER OF PARAMETERS SUPPORT (like SPL "OPTION VARIABLE")	NO	NO	YES+	YES-	YES	YES
NUMERIC FORMATTING/INPUT	YES-	YES-	YES	YES+	YES+	YES-
FILE I/O (see "FILE I/O" chapter for more)	YES-	YES	YES+	YES-	YES	YES
BIT ACCESS (see "OPERATORS")	NO	YES	YES	YES	YES	YES+
POINTER SUPPORT	NO	NO	YES	YES	YES	YES
THE ABILITY TO WRITE PROCEDURE-LIKE CONSTRUCTS THAT ARE COMPILED "IN-LINE", FOR MAXIMUM EFFICIENCY PLUS MAXIMUM MAINTAINABILITY	NO	NO	YES	YES	YES	NO
LOW-LEVEL ACCESS (ASSEMBLEs, TOS, registers -- often useless, sometimes vital!)	HNO	HNO	HNO	NO	NO	YES

REALLY NICE FEATURES TO PAY ATTENTION TO

Just some interesting things, mostly implemented in only one of the three languages. I just wanted to draw your attention to them, because they can be quite nice:

* PASCAL/XL'S TRY/RECOVER CONSTRUCT. A really nifty contraption -- see the "CONTROL STRUCTURES" chapter for more info.

* C'S "FOR" LOOP. You might think it's ugly, but it's quite a bit more powerful -- in some very useful ways -- than SPL's or PASCAL's looping constructs.

* C'S "#define" MACRO FACILITY. I wish that PASCAL and SPL had it too; it lets you do procedure-like things without the overhead of a procedure call AND without the

maintainability problems of writing the code in-line. ALSO, it lets you add interesting new constructs to the language (like defining your own looping constructs, etc.).

* SPL'S LOW-LEVEL SYSTEM ACCESS. Although you'd rather not have to worry about registers, TOSs, ASSEMBLEs, etc., sometimes you need to be able to manipulate them -- SPL lets you to do it.

STAFF TRAINING, WHY, HOW, & WHEN

Charles H.R. Volz
VOLZ Associates, Inc.
34 Undine Avenue
Winthrop, MA 02152

ABSTRACT

A knowledgeable staff is very important to any operation. However, few organizations have a systematic approach to training their staff. This paper and presentation will talk about why continuing education is important and its benefits. We will talk about what is available, how to choose from among the various options, and when to train. We will also discuss how to attract and keep staff with training.

1. IS IT NEEDED & WHY?

I have a quote from Susan Boyd of PC Concepts. She said, "Training is the investment you make in people so hardware and software pay off".

Staff training should be a continuing program for every company. Without a training program employees become stale, complacent, and frustrated. An employee's lack of knowledge can hurt an organization by costing money and time. Lack of training can waste an employee's time. It can waste the investment made in equipment, systems, and software.

Most people want to learn more about their job. They want to tackle new challenges. They want to do things correctly and efficiently. They want to be proud of their work. Training makes it possible for people to do these things.

For a company, employees who know their jobs thoroughly will be more effective participants in reaching the goals of the organization. It can cause morale to be and remain high. Employees are more willing to take on new assignments knowing that training will be provided. There is more efficiency. Goals are met or exceeded.

I. Benefits

A. Greater efficiency

B. More knowledgeable staff

C. Happier staff

2. WHEN TO TRAIN

When should a company train? For new employees when they start. No matter what level of expertise the new person may have all new employees should be given an extensive orientation. This orientation should cover department and company standards and procedures. The employee will then be comfortable with following them.

Employees coming into a situation where the equipment or system is new should be trained. The training should begin as soon as possible after the start date. On the job training should be discouraged.

All companies and departments are starting new projects or bringing in equipment. A lot of these things need the employees to learn new skills. Training is very important in this area for a smooth integration of equipment and completion of the project.

Another area where training is needed is when an employee's performance is poor. Training may give the employee the boost to get the job done efficiently and effectively.

Some employees ask for training. They have perceived a lacking in their own knowledge. Or they have glimpsed at a new technology that might benefit the organization. These requests should not be dismissed out of hand. Rather management should evaluate each request and act on its merits. The employee just might be right.

A final area where training is essential is when an employee is given a promotion or a new assignment outside of their current expertise. Training will help give the company the payback it is looking for with this change. The effectiveness of the employee will be improved if the employee understands what is being asked of him/her.

On the job training should be discouraged. This is the costliest way to train. On the job training is a very slow way to learn. The employee is frustrated and or discouraged. The employee feels unproductive and is unproductive.

If, per chance, on the job training does work, the employee learns enough to function adequately, most plateau. It takes a lot of initiative on a persons part to self train themselves. When this happens usually only what is absolutely necessary for the job is learned.

Are we saying that no self-training or on the job training should be done? No. There are times when this method is very appropriate. It depends on the individual employee. If allowed it must be monitored.

II. When

A. New employee

B. New situation requires it

C. Employee needs it

D. Employee asks for it

E. Employee is being reassigned

3. WHAT IS AVAILABLE

There are various method for getting employees trained. In our HP 3000 environment the biggest place to look for training is from Hewlett-Packard. Another place is the software developer whose package a firm is using.

For other public classes there are two more organizations to look into. The first is independent firms local and national. If on a HP related subject, many times these firms are more cost effective than HP or the large software suppliers. Most have worked with the subject matter for many years. They may even be more "qualified" to teach than the supplier. Non HP related seminar companies would be more generalized. But most of the instructors used are considered experts in their field.

The other place to look is colleges. For generalized training they can be excellent. Elapsed time is longer with a college, however.

Many of the same above will provide in-house training. The benefit of this type of training is that it can be tailored to an organization using their own data. Easy follow-up

and monitoring of employee progress are two other benefits of in-house training. For large numbers of employees it is also very cost effective.

Also for companies with on going internal training hiring a staff trainer is very effective. A staff person gets to know the staff and the company very well. Training can then be tailored to both the company and the employee.

One caveat though; treat an in-house training session as sacrosanct. Meaning do not interrupt unless sickness, death, or public safety forces you to. If you do the investment in training will be lost. And all lose.

In electronic training there are video, audio, computer based training, and interactive video a combination of video and computer based training.. Although generalized, these are self-paced; the employee can move forward when ready. Also time away from work is limited and the courses can be reused saving money.

However, with electronic training, an instructor is usually not available. As a result questions can not be asked. The hand holding support is not there to help smooth rough areas.

A final area for training is conferences. Much information can be learned from these meetings. Not just from the lecture sessions but also from the informal discussions during breaks. Also local/regional professional groups can offer a learning experience. (User groups, ASM, DPMA, etc.)

III. Training Methods

A. Public

1. Hewlett-Packard
2. Software Suppliers
3. Independent Seminar Houses
4. Colleges

5. Features - Benefits

    a. Generalized
    b. If Vendor related the product specific
    c. Can be cost effective for low count employee training

B. In-house

1. Hewlett-Packard
2. Software Suppliers
3. Independent Seminar Houses (Consultants)
4. Some Colleges
5. In-house Staff Trainer

6. Features - Benefits

a. Tailored to organization
b. Cost effective for training many employees at the same time
c. Easy follow-up of course value
d. Can monitor employee progress more easily
e. For in-house staff trainer tailored to organization and employee

C. Electronic Training

1. Video
2. Audio
3. Computer based
4. Interactive video

5. Features - Benefits

a. Self-paced
b. Reusable

D. Conferences & Symposia

1. Professional associations
   a. ASM
   b. DPMA
   c. Etc.

2. Computer user groups
   a. INTEREX
   b. Local/Regional

3. Features - Benefits

a. Massive amounts of information available
   (1) in scheduled sessions
   (2) during breaks

4. HOW TO CHOOSE & WHO TO CHOOSE

Training is only a good as the people providing it. The criteria listed below comes from research Lotus Development Corp. did prior to setting up their Authorized Training Center Program.

4.1. Price

It should go without saying; cost versus value needs to be evaluated to get the best return. It makes no sense to spend a lot of money on training if the people doing the training are not knowledgeable in the subject or are not comfortable making presentations. This also applies to the major software suppliers and HP. An inexperienced person should not be teaching others.

4.2. Commitment

Will the firm doing the training be around for the long run? (This is also important for software houses but that is another issue.) It is very important the firm doing the training not be a "fly-by-night" firm. The company needs to be committed for the long run. The company should have a proven track record for training.

For the in-house programs, again the commitment to stay in the marketplace is important. If the seminar is going to be tailored to the client, the client should expect the development of the course to take time. Also the client should insist on confidentiality.

4.3. Strength of Relationship

You should not be afraid to call on the firm giving a seminar to discuss content, ideas, methods, or subject. A good relationship will ensure better programs, more success for the employee/firm attending a seminar, and, of course, increased productivity.

If the relationship between the trainer and trainee is not good, the value for the attendees goes down. This is not to say that it is necessary to know the firm doing seminars or the instructor completely. Just that the opportunity to discuss ideas, etc. is there.

For in-house seminars, the willingness to customize a seminar to the clients environment may well show a willingness to develop a strong relationship. Customization may not be applicable to every course offered. But if it is, the use of a clients own data, environment, and/or vernacular will help them get the most out of the course.

4.4. Training Off- or On-site

For those attending a seminar it is usually better to have the training off site. This is true even for the "in-house" seminars. The reason is that the students are away from interruptions. They can concentrate on the subject being presented to them. It matters not whether the firm giving the seminar has there own facilities or use public facilities. For that matter space on the client site is acceptable if the client management refrains from interrupting. The only immediate reason for interrupting is for death, sickness, or safety. Any other interruptions should be thought out carefully before being done.

4.5. Timing

Naturally, the best time to train is before the need is critical. The scheduling of a training session should be made with this thought in mind. It makes no sense to train someone in tax preparation after the tax season. Unless it is in preparation of the next season.

4.6. Resources

Resources is a broad category. Here we mean class size and equipment for student use. Resources are also determined by how "hands-on" the class is to be.

The number of students per instructor depends on the type of program being taught. Very "hands-on" should be less than ten students. A very interactive session with questions should be kept below thirty. A lecture has no limit other than what the facility can handle.

If equipment is being used in the learning process, i.e. a PC or terminal, it is best to have a one-on-one situation, one student to one PC/terminal. That way they can learn from there own mistakes. Certainly no more than two students per PC/terminal should be allowed.

4.7. Product and Industry Expertise

The expertise of the instructor is very important. Is the instructor well known in the field? This does not necessarily ask if the instructor has published a lot of papers; but what is the instructor's reputation.

Length of service in an area is also important. But while evaluating the number of years experience, keep in mind how new the technology is. For example, MPE-XL classes will begin in another year. No one has more than two years experience with it. In this situation it is better to look at the instructors overall experience and if they have taught before.

Teaching experience is very important. One maybe an "expert" in a particular field but be unable to teach.

4.8. Levels of Instruction

While it is not advisable to have different level of experience in a class, sometimes it can not be helped. It is, therefore, necessary to insure the instructor can handle experience levels. It is also necessary to look at the level of experience the seminar is being aimed at. Prerequisites may preclude various levels from attending a class.

If multi-level experienced students are allowed and the instructor can handle them, it is necessary to have appropriate material for all the levels.

4.9. Materials

For any type of educational session, presentation materials should be made available. Vendor specific sessions should provide the presentation, tutorial, workbook and a reference guide. Additional support material such as glossaries, articles, and technical tips are helpful.

4.10. Follow Up

Follow ups run in many formats. The very least should be a willingness to send answers to questions that remain unanswered at the end of a session. The offer of a subscription to a newsletter or information reporting service is helpful. Also helpful is a follow up session to advance students along.

Another area in follow up is a willingness to ask for session evaluations and critiques. This is a very important indicator of a trainers commitment to a long term service.

IV. Who to choose

A. Price

B. Commitment

C. Strength of relationship

D. Training off- or on-site

E. Timing

F. Resources

G. Product and industry expertise

H. Levels of instruction

I. Materials

J. Follow up

5. HOW TO PICK A COURSE

5.1. Audience

Picking a course is as important as picking who to teach the course. The first criteria to look at is what audience the course is aimed for. This does need careful attention because the nomenclature used can be vague. Just as it makes no sense to send a person to an advanced system manager class if they have never been to a system manager class. It is equally wrong to send a person to a class titled "New MPE Issues" if it is really an intro to MPE. Above all, avoid the general good for all types of attendees. No one benefits from these classes.

5.2. Prerequisites

Audience prerequisites should be measurable. Actual skill levels should be stated clearly. Statements such as 3 months experience or if familiar with such and such are not clear enough. Measurable means can the attendee can perform at this level or do a particular task after taking the class.

5.3. Performance

The sessions performance oriented objectives should be clearly described in measurable new skills. The objective should list who learns, what will be learned. It should be explained under what conditions the student can accomplish the goals; what level of performance the student can achieve. Particularly helpful is a statement on how to measure for the goals.

Measurability will be shown if the class brochure states clearly these issues in action words rather than passive words. For example, words such as do or write are action words. A passive word is understand.

5.4. Support material

When choosing a course look at the outline of content. One should always be given. It may be concise. But the outline should be descriptive of the course content. If case histories are appropriate their content should also be described.

5.5. Practice and feedback

Practice sessions can be a very important aspect of training. Some classes by their very nature and brevity preclude them. But it may be something that fits your situation. If practice sessions are included check on whether testing is a self test or a proctor given test. Also check on whether the results of testing is immediate. Immediate test results should mean quick correction of errors.

Question answering is also important. Most questions should be answered immediately. If the question was ahead of an issue then reference should be made again about the question. Questions should be encouraged even in self-paced classes.

5.6. Review

We mentioned review earlier in terms of evaluation as a whole. Here we what is asked is if the material can be reviewed easily by the student after taking the class. If a case study is provided can it be used for later review.

5.7. Self-paced training

For self-paced training you need to know if the student can focus in on the material quickly after being away from it.

Also, the course should be designed in a modular manner to allow for frequent and easy review of material before going on.

5.8. Other

For on-site classes customization should be readily available.

V. How to pick

A. Audience

B. Prerequisites

C. Performance

D. Support material

E. Practise and feedback

F. Review

G. Self-paced training

H. Other

6. HOW TO GET TO A TRAINING SESSION

Justification is the key to getting to a training session. One must prove the benefit of going versus not going. If the training session meets the criteria stated above, there should be no problem in justifying.

For some other tips on how to get there. If a potential student proves to him/herself that the session is valuable they ought to just go. Invest in his/herself; take the time off and attend. The student will get the benefit and it makes no sense to deny oneself because an employer won't spend the money. (An employer might actually have more respect for the person for going on his/her own.)

Another way which is good when getting in new equipment or a new system is to build it into the purchase order. If possible, note the dates of the session on the PO so that all is approved at the same time. Having the training in the original PO avoids the chance of refusal.

VI. Getting there

A. Justification

B. Student invest in his/herself

C. Include it into PO for new equipment or system

7. WHEN NOT TO TRAIN

Giving classes out as an award for job performance or any other reason actually negates the reason for training. Everyone wants to do a good job. They want to grow. Training should be looked at as an investment with benefits and long term saving being reaped by the company.

To give someone training in an area in which they are already skilled makes no sense. And yet it is done by well meaning people. There is no benefit to the company or the employee. It would be better to just give the trip without the class and call it an incentive award.

It could be argued that sending a person to conferences and local/regional user group or other professional group meetings should be an award. And sure there are social activities at these meetings and conferences. But there is also valuable information available which when brought back benefits the company. At the very least a rotating schedule of employee attendance should be arranged so that all can grow. This also ensures that the company gathers all the information from all points of view.

VII. When not to send

A. As an award

8. CLOSING

Training is a luxury no organization can afford to be without. Training stimulates employee growth and company growth. Training causes a synergism by the sharing of ideas and knowledge. An exponential factor comes into play when there is an on going training program benefiting all as a group much more greatly than by themselves.

The education process must never stop. It has a much shorter life-cycle in today's fast changing technical world.

ABSTRACT

Logical Data Base Design

Mark Wallace, Robinson, Wallace & Company

There are two essential aspects of defining the user's requirements for a new application. One is to define the functions to be performed, and the other is to define the data that must be stored in support of those functions. In the past two years, I've talked about Essential Systems Analysis as the method of choice for defining functions.

This year's presentation will survey the fields of data analysis and logical data modeling. Data analysis helps us identfy the data elements (or items or fields) that must be stored. Data modeling helps us represent those elements as attributes of logically selected entities.

The resulting logical data model can then be used for physical database/file design. In the design step a plan will be drawn to implement the model as some combination of actual computer files. On the HP 3000, these could be HPSQL, IMAGE, KSAM, MPE, or RELATE/3000 files, for example.

Key topics to be discussed include, entity-relationship modeling, relational modeling (normalization), data modeling in the system life cycle, transition to physical data design.

You Gotta Do What You Gotta Do

Craig W. Wallin

IRECO Incorporated
11th Floor Crossroads Tower
Salt Lake City, Utah 84144

Why This Paper?

We are relatively new Hewlett Packard users. We have had our equipment almost two years. Last year three of us attended our first Interex conference in Detroit. It was a learning experience. Listening to what other people had been through prompted me to write something about our own situation. Not because we represent anything unique. Not because we discovered any earth-shaking revelations. Basically, I'm writing this paper to tell the same old story with a different setting; to reiterate the same key points...stuff that's just common sense. I hope that reading it or listening to it be told again is comforting. Give hope that we're all really not screwed up. We're all out there doing what we're paid to do. Sometimes the right way. Sometimes just to get by. Doing what we've got to do. Maybe, in re-hashing it one more time, we can be reminded of a perspective we'd lost track of. Or maybe, with any luck, we'll learn something.

IRECO Incorporated: The Company

Ireco Incorporated, an international explosives company, based in Salt Lake City, Utah, has operating facilities in 26 countries, on every continent. In addition to wide-spread Company operations, Ireco has also established licensee relationships with foreign manufacturers in key areas of the world. Ireco is owned by Dyno Industrier A.S. of Oslo, Norway. Dyno is one of Norway's largest and most long-established companies. It's history goes back to 1865, producing dynamite on the basis of Alfred Nobel's patents.

Ireco's full spectrum of products and services include: bulk and site-mixed explosives; packaged explosives: slurries, gels, emulsions, and dynamites; boosters: detonators and detonating cord; defense systems; seismic exploration explosives; nitrogen products; equipment design and manufacture; and sales distribution.

Data Processing/MIS

How far can a company come in the world of data processing in an eight year period? Of course there are alot of factors that determine any kind of an answer to that

question. It develops as much as it has to, because you gotta do what you gotta do.

In 1979 Ireco had an IBM System 3/Mod 10. The staff consisted of two key punchers, two operators, one programmer, one analyst, and a department manager. After a major implementation of an inventory software system, it was determined that the System 3 was pretty well maxed out. It was time to begin looking for new hardware. In addition, the philosophy of data processing was really changing from batch to on-line. It was time for Ireco to attempt to catch up. After extensive analysis both internally and by an outside accounting firm, a decision was made. In August, the new vendor's equipment was installed and the conversion began. There were all kinds of software problems. Because we had been on a System 3, every program we had was written in RPG. Several hundred of them. The machine we were going to had an inferior RPG compiler that we seemed to be debugging. Our inventory system's main files (part master, bill of material, etc.) relied heavily on the use of relative record numbers. Relative record numbers were not supported on the new system. Pain, woe, and agony.

There were hardware problems. Everything imaginable went wrong. Engineers came and went. CPU's came and went. Salesmen and district managers would meet with us, agree that the situation was intolerable...fly away and never be heard from again. The machine, when decided upon, had been billed as the "machine that would never go down." We had to eat those words every day.

Almost a year later, the conversion finished. We made it work because we had to. Don't get me wrong. We were further ahead than we had been. There were some inherent benefits. For one thing, we could have more than one job going on. Those of you old enough will appreciate what I'm saying. Those who had to schedule any compiling and testing around production, which in all honesty, was a bitch. At any rate, the move gave us capabilities that we never had before. Even though it was an on-line system running mostly batch, we did have concurrent access, the ability to inquire, etc. We began to evolve. New software was developed and acquired. The staff grew. The information became more timely and reliable. Unfortunately, the hardware never did.

In late 1984, issues were coming to a head. Several major software purchases were made in a bundled deal, but the vendor was backing out on converting two of them because of problems with our hardware vendor's COBOL compiler. They also announced that the two applications they did have converted and running on our machine were going to become unsupported products. These facts, coupled with our long history of down time, led us to begin a search for new

hardware. With a staff of six people, consisting of a manager, three programmers, system manager, and an operator, an extensive hardware study was undertaken. In February of 1985, just when we thought we had considered everything possible and had recommended Hewlett Packard as our solution, our studies took on an added challenge. Ireco signed a letter of intent to purchase the Explosives and Nitrogen Products Division of Hercules Corporation, a move that would more than double the size of the company. It was time to rethink our decision based on an entirely new set of criteria. Criteria that we could not completely define or understand.

The Acquisition

We believed there were four major areas of concern: 1) The new hardware had to be able to accommodate Ireco's expanding data processing and management information needs both now and in the future. We had to build a solid foundation. 2) The new hardware had to allow us to realize our investment in the software packages we had acquired, both implemented and yet-to-be implemented. 3) The new hardware had to have a proven position in the computer marketplace as a reliable system. The vendor must have a track record of excellent service and support. 4) The new hardware must have a large selection of software solutions, especially in the manufacturing and payroll/personnel areas. These were two systems we knew we would have to replace immediately.

Networking also became a concern. With the Hercules acquisition we would have the need to communicate with plant and product distribution sites at Port Ewen, New York; Donora, Pennsylvania; Paintsville, Kentucky; Louisiana, Missouri; Carthage, Missouri; and Bessemer, Alabama.

Mentally we were on the rack. We were making one hell of an important decision. And we'd been living with a bad one that had been made six years previous. The bright spot in all of this was that we were being afforded an opportunity few in our profession ever get. In essence we were going to start over. Some of our biggest messes were going to be abandoned with the old hardware. Realizing the magnitude of impact the acquisition was going to make, we knew most of our systems would not be worth converting. The hardware decision had to be made so the software determinations could begin.

After much additional study, we felt that we would need to start out with centralized processing, but we also felt that we could not avoid distribution of processing capabilities to handle future information demands. Again Hewlett Packard was our choice, based on a plan to acquire multiple machines over a five year period of time. Our reasons could be summarized in saying they were the best fit for us, our

situation, and our uncertain growth potential. We began a partnership. We were ready to tackle the next decisions.

The acquisition took place June 10, 1985. As part of the closing agreement, Hercules allowed access and use of their computer and certain applications for a year. That would save chaos at the plant sites until we were ready with replacement systems. Files and data were separated. Hercules became essentially a service bureau for us until we were ready to incorporate the functions within our new (yet to be determined) software. A Series 37 was installed in Salt Lake City so that communication could be established with the Hercules systems in Wilmington, Delaware. The MIS group assumed responsibility for the Delaware link, began it's HP orientation by utilizing the 37, and maintained the applications that were still being run on the "old" system, accessed by what we termed "old" Ireco. Plans were underway to increase our staff. By July our programming staff had grown from three to seven.

Hardware's Decided, Software's Next

In the software arena, we were performing in two rings. In one, new Sales/Manufacturing software had to be selected. A project team was formed. A Request for Proposal was written and sent out. Understand that this was not a simple task. Writing an RFP requires a knowledge of the functions the software is to fulfill. The acquisition of Hercules introduced new areas of manufacturing that we had little or no experience in. In fact, we now had to accommodate the needs of multiple types of manufacturing: job shop, process, continuous, and discreet. In the other ring, the immediate need was payroll. The old Ireco payroll system did not have the flexibility needed for the addition of the Hercules employees, the expanded benefits package, and union contract considerations. Wage-roll employees at each of the new sites were being handled at their locations. Old Ireco was on old Ireco. The immediate need was the Hercules salaried group. Again an RFP was written. Vendors and products were evaluated. A decision was made and we were off and implementing. I want to concentrate on the payroll project for a few paragraphs because I think it represented, in one application, exactly what was going on with every application that make up our systems.

Getting that first payroll out was a real accomplishment. In less than a month, we had evaluated and chosen a new package to run on our new machine (knowing it would only be temporary on the 37); gathered, organized, and entered the necessary information; written the necessary user exits required for our "exceptions" to the generic package; tested; and above all, met our deadline. The size of the initial payroll (about 350 employees) was ideal for our orientation to the system. When the 68 was installed in

August, we were ready for phase II of the project, which was to get old Ireco off the old system and on to the new. This was more complicated because of the quarter-to-date and year-to-date information that had to be transferred. A conversion program was written to extract as much information as possible from the existing system to transfer to the new. In September another 350 employees were added to the new payroll. The project group kicked back. We thought we'd made it for a while. Ha. We were then told that the corporation wanted to be able to have all the wage-roll people on the system by the first payroll of 1986. We knew that was inevitable, but we thought we'd have time to breathe before researching the task out. The complication was that most of the plant sites were unionized, and each of the unions had unique requirements when it came to earnings, deductions, etc. We were depressed. How in the world would we pull this one off? We'd met our deadlines thus far, but could we meet this one? Morale was lower than a snake's belly in Death Valley.

What we had happening on the manufacturing side influenced the approach on this one. We had again chosen Hewlett Packard as our solution, and in our organizational discussions they had told us that the only way we would meet our deadline was to recognize 1) the unnecessary, 2) the have to have's, and 3) the nice to have's. The unnecessary could be thrown out and ignored. The nice to have's had to wait. We had to focus on the have to have's. Applying that same philosophy to payroll, no matter what the gunk in between, the basic goal of any payroll is to get the employee paid. Paying an employee involves accumulating earnings to a gross amount, subtracting out deductions, and producing a check for the net. Pretty simple. Applying that: even if we couldn't duplicate the different earning methods required at the sites by the first of the year...they could enter the gross. Their manual or automated systems had been calculating that much, anyway. Even if we couldn't meet all the unique requirements as far as deductions, exceptions could be dealt with. There was a method within the system for them to be entered, also. In other words, we lowered our sights as far as what we were shooting for or trying to accomplish within that time frame. We made it more reasonable, more obtainable, and we did it. The first payroll run of 1986 produced checks for the entire corporation. We had added 700 more employees to the system successfully in less than a three month period. Now the effort was concentrated on accommodating the earnings accumulations we could not automate up front.

Back to manufacturing. We selected a software package that was flexible and could **easily** be made to service each of our different manufacturing entities (salesmen make everything sound so easy). The decision having been made, the software was loaded on the system in November. January

we were to begin site implementation. It had been decided that we would concentrate on the Hercules sites first (in order to meet the June 10 anniversary deadline), and then we would implement old Ireco. In order to oblige the individual site's requirements, it turned out we would need five copies of the software, each requiring individual customizing and programming. Complicate the need for multiple copies of the manufacturing software with the need for only one centralized order management system. HP had just reached agreement with a third party company to market their OM product as part of the manufacturing software solution. At the time we purchased the software, an interface between the two had yet to be written. We had been learning the system on an old software release. The new version, with the interface, was delivered in February. We ran a parallel test of the two plant sites we were implementing first. After entering two to three hundred orders and spending countless hours on the phone, we reviewed the results. 100% correct. We thought we were on track and retired to the bar. Going live at the first sites, however, the users found holes in the software. And then some. To make a long story short, the interface could not have been much worse. The implementors were only one jump ahead of the users as far as knowledge of the system. Inventories were inaccurate. We were ready to throw in the towel. But we didn't. We huddled, re-thought our plan, and came out fighting. What a nightmare! But we knew we had to start with the basics. We were billing the project as having three distinct phases: Phase I was the elimination of the Hercules connections. Phase II was the elimination of the old Ireco hardware and systems. Phase III was for re-examination and tailoring. Realizing we would be forcing a fit in Phase I, we hoped Phase III would even things out.

Anyone with experience in these kinds of situations, realizes the enormity of the tasks we were undertaking. We had just selected the software in November. We were beginning to implement in January. Did we have **what** for brains? How could we possibly know what we were doing? How could we have possibly reached our comfort zone with the products? Realize also that at the point of implementation, it was decided that we would go to a new corporate chart of accounts. From the MIS perspective, we were dealing with a new computer(s), new software, new policies and procedures, and new people. And we had five months to bring six sites on board. Say a prayer for the dudes with pocket-protectors. Also remember that we had put together a network, something that we were pretty naive about. We had put together the best dog and pony show we knew how, consisting of two 9.6 multi-drop lines. We felt like we had accomplished a lot when we finally got all six sites communicating with Salt Lake. But as the network began being used, the ponys soon died and we were left with the dog. We needed help and we

knew it. More on that later. Meanwhile, the battle for the June 10 deadline raged on. There were many who shook their heads and doomed us to failure. There were many who said we were crazy and doing things all the wrong ways. But we proceeded. We knew we'd get it done. As we saw it, we had to.

There's something to be said for honesty...and being realistic. We went to the first site, and a month down the road, it was frustrated and hostile. In retrospect, we decided we'd spent too much time discussing what was to come instead of what was. We modified our approach. We stressed that things were not going to go smoothly. We urged channeling of frustrations into helping us resolve the thing (or things) that were frustrating. We felt mistakes were only mistakes if we didn't learn from them. God knows, we made a lot of mistakes. We also learned a hell of a lot.

In the meantime, as the manufacturing and Payroll/Personnel projects were forging ahead, other issues were becoming priorities. The assets acquired from Hercules, numbering well over 1100, needed to be accounted for by year end. With the end of 1985 approaching, we had to implement a new Fixed Assets system. An outside valuation firm had been contracted to identify and value each new asset at each of the new locations. This valuation was completed and received by Ireco in mid-November. We only had about five weeks before year end, but our first milestone was reached. It fulfilled the immediate need. Much more work, and the conversion of old Ireco assets from the old system were still ahead.

The financial systems were becoming a priority also. Beginning implementation of Accounts Payable and the building of the new chart of accounts on the General Ledger side began in mid-February, 1986. (Imagine the mess we were going through at month end. The accountants were consolidating information generated on three different computers, within two different application systems, from three totally different charts of accounts.) The first site was brought onto the AP system for March business. Another site was added each month until all the Hercules sites were on-line for June. We achieved our anniversary deadline. We'd provided the sites with the basic business [illegible]tions. We cut the Hercules cord.

Where We're At Today

In August it will be two years since our 68 was installed. Remember the five year plan for hardware? Our aim was right on but our depth perception was off. We blew that plan to hell in less than a year. The 37 has become a Micro 3000 and is the host for the Personal Productivity Software. We are in the process of implementing an office LAN. The 68

has become a 70. We have added a Series 52. We now have a Series 58 in New York.

As the software implementations progressed, it became apparent that our current datacomm configuration would not be sufficient to handle the load. Our network at that time, as mentioned previously, consisted of two 9.6 multi-drop circuits and one 9.6 point-to-point circuit (see Illustration 1). Since the whole realm of data communications was new to us, we decided to get some training and enlist the help of a couple of HP datacomm experts in defining what our needs were and how best to handle the load. This began a six month study into the transaction processing volumes from each of the sites and the proposal of a datacomm plan to handle these volumes and maintain acceptable response time. This proposal consisted of a backbone x.25 system-to-system communication network with a terminal network laid over the top (see Illustration 2). All of the circuits were changed to a point-to-point operating environment. The speed on some of the circuits was boosted to 19.2 so that each site has it's own 9.6 slice. The new network also built in redundancy and contingency capabilities, accomplished through dual dial restoral modems and circuit re-routing capabilities. The new network was implemented in February of 1987.

Payroll/Personnel, Order Management/Accounts Receivable, and Fixed Assets have been implemented. Accounts Payable and General Ledger were converted. MM (Materials Management) has proliferated. There's a **MMAN** (ammonium nitrate), **MMDY** (dynamite), **MMPE** (Port Ewen), **MMCA** (Canada), and **MMWJ** (West Jordan). PCM (Production Cost Management) is in the final steps of implementation. In the wings are PM (Production Management), PO (Purchase Order), and potentially, new financials.

And what about our users, you ask? Somewhere in America is there a user community singing our praises? Guess again. There are frustrations, especially among the former Hercules people. They had been users of systems that had evolved over years of effort. Starting over for them was no picnic. But we have said all along: **How do you eat an elephant?** A bite at a time. We could throw in the old **you have to crawl before you can walk.** There are any number of phrases. The point is the same. We had to start over. We selected software that we'd determined was as close a match to us and our business as we could possibly get. But there's never a perfect fit. We began an undertaking that will take years to solidify. Most users have no appreciation for that. They don't see the over-time and the burned out bit-twiddlers. What do they care about alcoholism or divorce? They just see their screen.

There has been some talk that HP might use Ireco in an ad. We pictured ourselves in towels, hopping over floor buffers; racing to the telephone to shriek, "What if..." My personal fear is that if we ever are used, Ireco people will see the commercial and not recognize they're seeing their own company.

SUMMARIZING THOUGHTS

Management Committment

We were consistently told that we would never succeed without management committment and involvement. That's probably true. Ours just never came in the textbook manner. Both needs were fulfilled indirectly. It was an extremely busy time period for most people in the company. We adopted the philosophy it was easier to get forgiveness than permission, and forged ahead. Management was kept abreast of our plans and progress. We decided to involve them if we needed to bring in our big guns. We never needed to.

Times were tight for the company. Not only were they dealing with an assimilation and consolidation, but down turns in the seismic and mining industries took their toll on the explosives industry. Revenues were down when our expenses where high. But our Vice President of Finance put our task in perspective with the comment, "There is no more important time for the information to be good than when business is bad." He exemplified the importance of what we were trying to accomplish. He amply gave us his commitment.

Partnerships

You can't do it alone. There were partnerships that we established that continue to serve us well. Within the company, the added resources of the accounting department helped immensely. (I can't believe I wrote that.) Outside, the relationship with our hardware and software vendors was of critical importance. When we'd discover a bug, we were usually in the situation where it was affecting business operation. We were implementing live. If it didn't work we were hurting. We had to rely on our vendors a great deal. We didn't have enough experience with any of their products not to. (I think that situation is now different.) It was nice to never have had that sick feeling of abandonment. Our vendors realized we were all in this thing together.

Outside Expertise

If you're like us, you're a limited resource. Therefore, when faced with challenges you know you'll never have time to come up to speed for; if you can, get outside help. I'm not an advocate of "experts" who come in and tell you what

to do. I do strongly respect "qualified" opinions. Hire someone who knows, and have them give you more than one alternative. The final decisions should still be yours.

Our best example of this was in regard to our network. We were over our heads. We gave it our best shot in the beginning, but it was obvious there was more to it than we realized. When you're looking to get the most bang for your buck, utilize some of that expertise someone else paid their dues for.

Planning

How can you get anywhere if you don't know where you're going? The specifics of what we do are really almost planned day-to-day. You know what general direction you're heading, but you never know what turns you'll take or what fires you'll have to put out. You have to be careful to not be eaten by the detail, however.

We have been in some pretty ugly meetings in the last two years. Our endurance has been buoyed by the fact that most of the complaints have answers...and we know it. It's just a matter of time until we get to that particular implementation, enhancement, or whatever. The solutions are there. If you didn't have that in the back of your mind, you might as well pack it in. It's just a matter of time.

Another key to be mentioned here is **flexibility.** As long as our profession is service oriented, we'll be driven by the customer or user. If we get p.o.'d because they change their minds, we've lost sight of what we're all about. (I'm starting to get philosophical. Gag.)

Success is found in having a plan (both short and long term), staying focused (which is damn hard to do), and being flexible. Know where you're going. Mellow out so it won't kill you.

People

I saved the best for last. You can have the best computer ever made. (I'll let you argue over which vendor provides that!) You can have the most fantastic software ever written. But if you don't have good people, you don't have squat. That's true on both the user and the support side. And since most of us here are on the support side, I'll focus on that.

We could not have accomplished what was accomplished without the dedicated efforts of the individual project teams. We were successful and continue to be successful not because of any one person. It has been because of the individual strengths that make up the collective "**we**". That is what

carried us so far. That is what will carry us into the future. There were long hours. There was a lot of travel. There was a ton to learn and be responsible for. And it must not be forgotten that in the back of everyone's head there was this nagging little voice screeching, "You're doing everything wrong."

If you were citing the textbook methods of system implementation, we **were** doing everything in the "don't do it this way or else" section. But if you're lucky enough to assemble a team who care not only about the work they do, but about each other...there's no telling what can be achieved. Especially when you gotta do what you gotta do.

Illustration 1

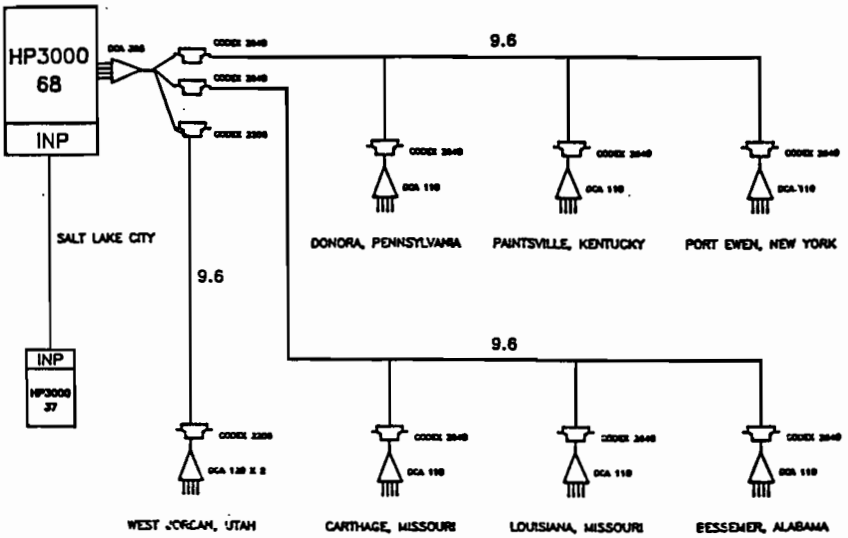
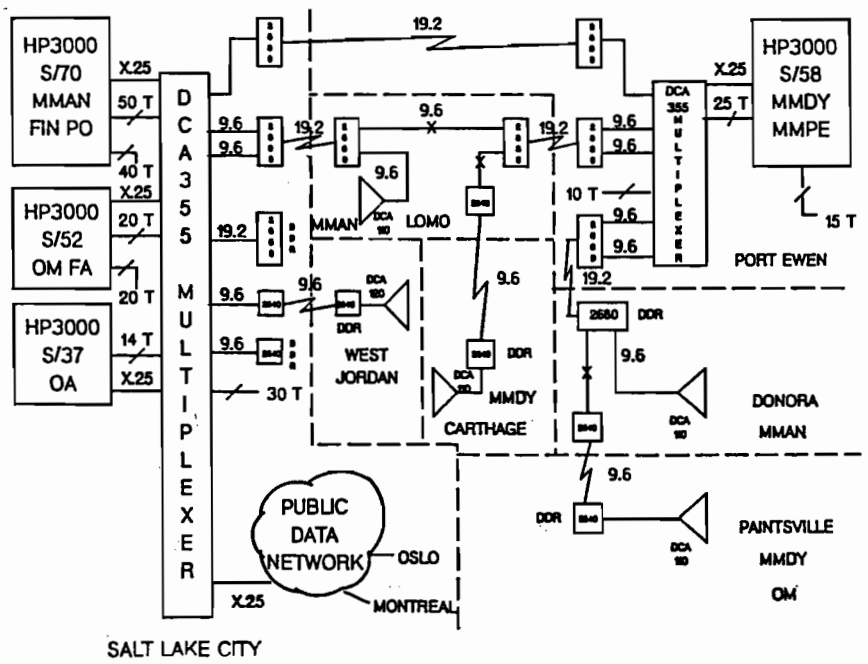


Illustration 2



INFORMATION CENTERS AND INFORMATION ACCESS

Fred Waters
Hewlett-Packard Company
8010 Foothills Blvd., Roseville, California 95678

INTRODUCTION

In a competitive world where superior products, services, people and even working environment are not enough, companies are constantly looking for ways to get ahead and stay ahead.

Information has become a key weapon in competitive wars. Executives who have information placed in their hands, when they need it, in the form they need it can make better decisions. These executives and their staffs can better understand their business, better control the processes of their business and better determine the direction their companies should take. Systems which can provide a foundation for such an advantage are in demand. The information center concept can provide such a foundation.

What is an information center? How can it benefit me and my organization? To answer these questions requires a look at where business has been historically, what has been done and what can now be done. And a little bit later in the presentation, to help clarify the issue, we'll take a look at a case study: a marketing department in a well known computer manufacturer. But first, a look at where business thinking is today.

CHANGE IS AFOOT

Executives are re-thinking the assumptions surrounding the information used in their businesses:

One result has been to regard information as an asset.

Another result has been to define information not simply as data, but as a combination of data, context, meaning and relevance.

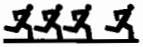
Another result is to upgrade the role of MIS from a support/staff activity to a "line" function.

And perhaps one of the most profound results is the recognition that there is a difference between transaction oriented and management oriented information assets.

These new perspectives have focused attention on using the information assets of a company to achieve four major objectives:

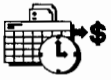
The Business Need

MANAGE THE INFORMATION ASSET



TO GAIN/MAINTAIN THE COMPETITIVE EDGE

TO ACHIEVE THE BEST ROI/ROA ON UNIT OF INFORMATION



TO ENHANCE PROFIT/CONTAIN COST

TO CONTROL THE BUSINESS

INTEREX2

But the question is "HOW?". The search is on to find a way of providing the information needed. The elusive goal is to have:

- Management Oriented -

THE INFORMATION NEED

- ***A rational method of collecting, managing, and distributing information which gives us a clear focus to:***
  - *Make better decisions*
  - *Expedite action*
  - *Maintain quality*
  - *Evaluate results*

On a company wide, sub-entity, workgroup and individual basis

---

INTEREX3

TRANSACTION DATA VS MANAGEMENT INFORMATION

A general recognition of the differences between transaction processing and management/professional information assets is evolving. When the management/professional information is pooled and looked at as a single entity, logically related, with context and meaning added, it becomes an information center.

In the transaction environment we are familiar with action triggers generating actions and ultimately a desired result. As an example, for an office supply company, the TRIGGER may be a customer placing an order. The ACTION may be the generation of a pull order, the generation of an invoice, a shipment and an open account with balance due in accounts receivable. The RESULT will be a closed transaction with the receipt of payment.

Transactions systems provide much of the foundation information used by information centers. Because of this, they share many common characteristics and especially the data itself. But there are also characteristics of transaction data which are unique:

Closure - A transaction operates in a closed loop environment (an order for example).

Data Oriented - A transaction focuses on one thing or one event. It will not typically make or break a business.

Transaction Terms - the data is defined in terms of the transaction only. For example, an order or a shipment. Historically, most systems, have used coded values to define the information more concisely.

Custom Solution - although any given industry has many common needs, virtually all businesses have customized transaction oriented systems which match the way those companies do business.

Process Flow, Time Fixed - The transactions represent the flow or business process. They represent the systems supporting the delivery of goods, services and so forth, to the customer.

On the other hand, management/professional staff oriented information is typically summary, trend, or historical information selected in the context of decision making, expediting action, quality and/or result monitoring. The information need starts with the business assumptions driving the objectives and follows on to the metrics, feedback and

actions. Each step requires information to properly assess what needs to be done. This type of information is usually characterized as:

Open-ended, Recurring - Closure doesn't occur and the need continues as long as that business opportunity exists.

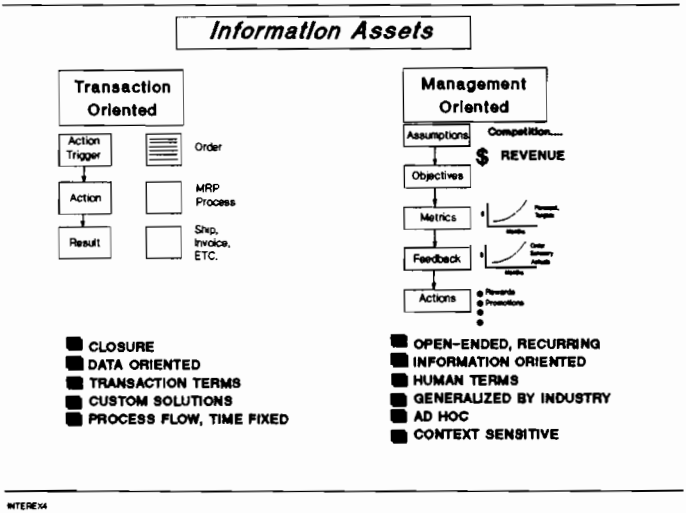
Information Oriented - The data has context, meaning and relevance.

Human Terms - The data is defined in terms that are easily understood by people, rather than codes or multiple definitions.

Generalized by Industry - Although each business's management information is varies according to business assumptions and objectives, overall management information needs aren't that different within a particular industry.

Ad Hoc - New needs arise, competitive situations change, ... all these require new responses, often very quickly. In addition to standard reporting, ad hoc access to information is essential.

Context Sensitive - Managers must define the views of the information they need in terms of the situation at hand.



INTEREX4

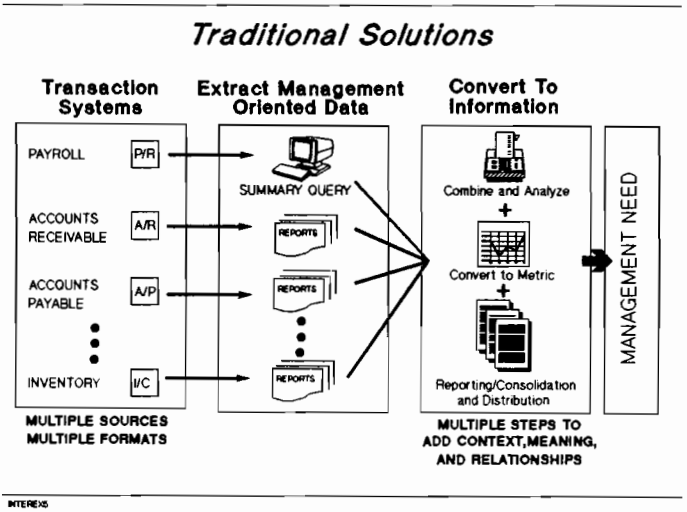
TRADITIONAL INFORMATION CENTER SOLUTIONS

To meet this management and professional staff need, companies created information centers.... although most of us didn't call them that. But none-the-less, that's what they were. They were collections of information, in various forms, at various locations. Managers who needed the information had that information collected from those sources which often consumed precious time. Or as an alternative, they made decisions without the necessary information.

The traditional solution was to compile extractions from various sources, combine the data, compare it to the metrics chosen, and create a report that summarized the resulting point of view.

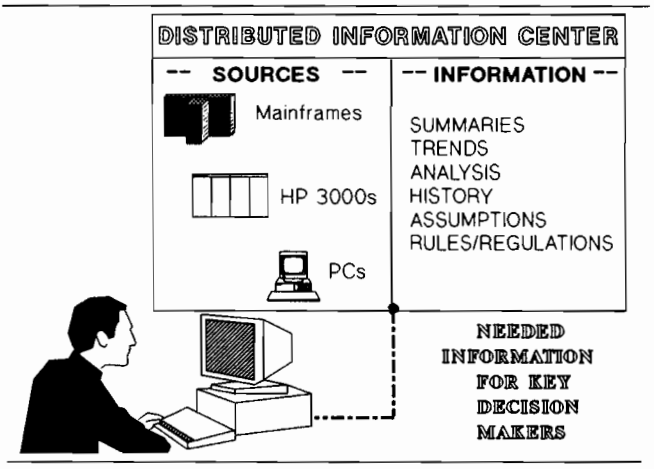
These traditional solutions drew upon multiple sources of information, in multiple formats. The desired information required multiple steps to add the context, meaning and relationships.

Such solutions are usually paper based. They are narrow in scope, focusing on one or two key application areas only. More over, they are time consuming.



What is needed is some way of allowing managers and others using management oriented information to reduce their efforts through a flexible, easy-to-use solution. The desired solution is an information center without the drawbacks of the traditional solutions. The desired solution needs to

span all relevant sources. It must allow MIS to maintain the prudent business controls over the information asset. The desired solution must present the users of the information, the information needed, when its needed and in the form needed. It must be cost effective. And it must be easy to set-up and easy to maintain.



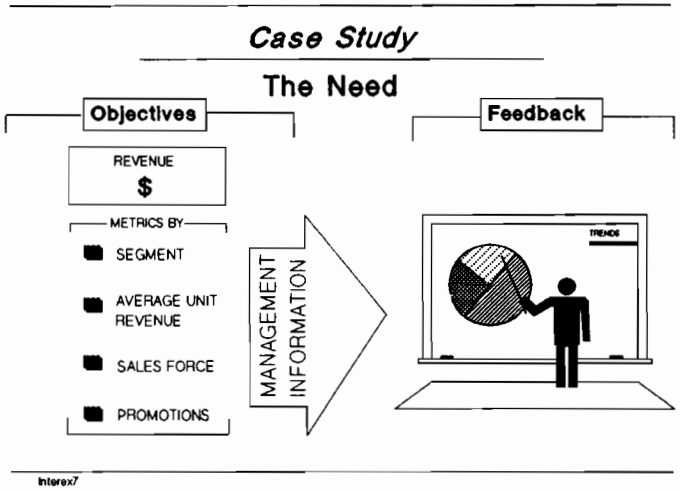
Interex6

AN INFORMATION CENTER AND HP ACCESS - A CASE STUDY

What follows is one example of how an information center solution was implemented in the marketing department of a computer manufacturer. This information center is a "subset" of a larger information center which supported several functional areas (finance, marketing, and so forth). For simplicity, we'll concentrate on that portion which was used by the marketing department.

The marketing department has the need to understand many of the business facets of the products they release. One facet of particular interest has been to understand if the discounting structures in place were realistic given the companies profit objectives.

This means defining the term "realistic" as a metric. Further, collecting the profitability and trend information, and filtering out special promotions is required. It means looking at the appropriate market segments and associated sales forces. Potentially, also, contracted support data may have to be taken into account. Finally, an analysis with appropriate graphics is needed to justify any actions recommended.



What had been done in the past was a laborious process. The marketing department would have had to sift though line item after line item to understand the order information. Codes reflecting discount structures, market segments, and other pertinent information would have to be interpreted and sorted. If contractual information was need, it meant going to another group and getting reports. Tables of prices might be needed.

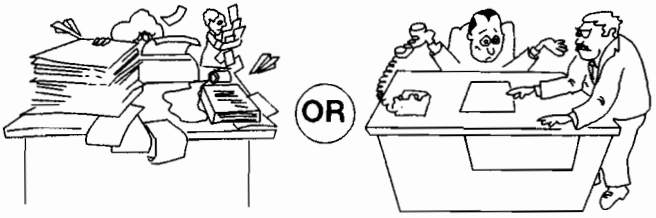
Similar queries for information from reports and multiple sources took days, at best, to address. Often, it was simpler not to perform the request, since it was too late and of little value to the requester when received. Time had made the decision for them.

Often too, another iteration would be required from another perspective. Perhaps a different market segment had to be considered. Managers or staff requesting help from MIS took their lives in their hands. MIS staff felt they were letting their users down. The result: a no-win situation.

The professional marketing staff often responded to their customers (their bosses and others needing the data), with "I'm working on it", or "I dunno".... Basic business questions often had to be answered with educated guesses which in some cases proved quite wrong.

Case Study

What Had Been Done



"I'm working on it" "I Dunno"

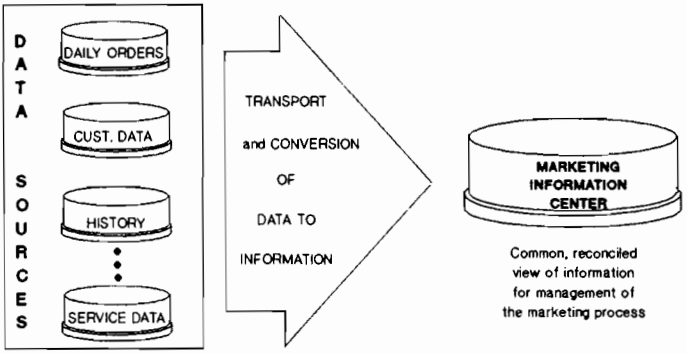
interex8

What was needed was an information center. This center would "logically pool" all the relevant marketing information. It must provide a mechanism for combining the summarized daily orders, customer data, history information and service data.

This center needed to provide the transparent glue which cemented the sources together. It needed to reconcile differences in data types and resolve communication paths. It also needed to have the flexibility to re-iterate views of information from new perspectives.

Case Study

What Needed To Be Done



interex9

With the help of MIS, the marketing department chose to implement HP's Information Access.

Information Access has within its standard capabilities the ability to look at transaction databases as the foundation for an information center. Information Access can provide the functionality required to translate transaction data into information while maintaining and even enhancing the security of the data.

In the case of the marketing department, several datasets were easily mapped into the information center (e.g. history data, service data...).

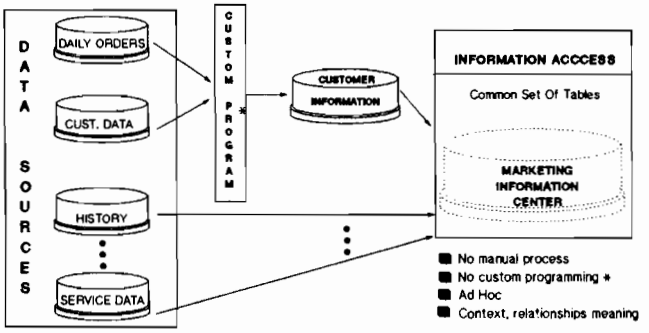
It is not always desirable or prudent to have direct access to transaction databases, however. As an example, for security reasons or to minimize performance impact on high volume transaction databases, access may need to be limited to certain times. Additionally, it may be best to create information center information at certain points in the daily processing stream.

Because of this, the MIS department elected to use a custom program to create another set of information for the information center. This program was written in COBOL. Today, however, it could also be configured in Information Access View Table form and incorporated with the Host Batch capability.

From the view of MIS, the implementation required one custom program and the appropriate dataset definitions within Information Access.

Case Study

Information Center Implementation



*SEE TEXT FOR DETAILS

interex10

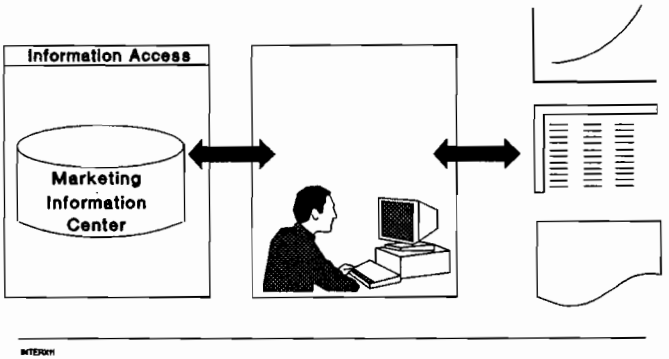
From the standpoint of the marketing staff, the information available appears as one common pool. Information from all sources is presented as tables available for access.

Marketing staff can now easily access the required information. In the case of our example, a join and summary of order data by market segments produces an intermediate result. This intermediate result, still contained within Information Access as a saved table, was then joined to relevant support and customer data. The final results were output to a spreadsheet, a graph and a report.

These outputs were combined later in desktop publishing software to create a single document.

Case Study

INFORMATION CENTER USE



CONSIDERATIONS

Before looking at the results, and summarizing the benefits, it might be helpful to understand some of the trade-offs that were made in choosing Information Access as the tool to implement an information center.

Information Cost

Perhaps the biggest consideration is the value of the information versus the cost of getting and maintaining it. This is an extremely hard item to quantify. It means looking at what information people use, what they need and what it would take to get it.

To help in the process, virtually, all information was made accessible. Some views (reconstructions for a particular need) were used. By consulting with marketing staff, the MIS department made adjustments to these views and to the types of foundation information available. It should be noted, this was done only for clarity (reducing the number of information items from which one could select).

As a metric, it was found that for an average workgroup of ten individuals, three databases are used. The analysis time for MIS is about 4 1/2 days. This time is spent in determining the specific needs of the end-users. Then an additional 1/2 day is spent defining the requirements in Information Access.

To this time, the time for custom programming, if any, would need to be added.

In the case of our example, the fact that the marketing department was able to act a projected 2 to 3 months earlier resulted in significant cost savings. By even conservative estimates, the implementation would pay for itself after 2 or 3 such decisions.

Information Need

Another consideration is the selection of who gets what information when. The decisions surrounds this factor were relatively easy for the marketing department. Everyone was given access to all the marketing information. For other uses, however, MIS working with the user would, as part of their analysis, need to look more carefully at this. Trade-offs would need to be made between what the users needed and what could be made available to them. How much detail information would be needed for an upper level manager would be yet another question for resolution.

Equipment

In the case of the marketing department, all networks and computer systems were in place. For new implementations, some sources may need to be added to the network. Additional cost would be associated with this. It would require a trade-off analysis on the value of the data versus the expenditures.

Mind Set

An all too often overlooked consideration is the mind set of the organization. An organization may not be ready for an information center - that is it may still be too busy dealing with the transaction world.

Typically companies that are smaller, or perhaps even a start-up group in a larger organization, may not have the scale to justify an information center. Supporting transaction systems may not be in place yet for example. Individuals working in such an environment may be focused on transaction needs.

For the marketing department, the scale to justify an information center was reached in its 3rd year.

Case Study

IMPLEMENTATION CONSIDERATIONS

TRADE OFF:
Value of information
vs.
cost of getting and maintaining it

Are the necessary computers in place?

Who in the organization needs what information at what detail, when?

Does the organization have the right mind set?

INTERX12

THE RESULTS

Several results were achieved and are continuing to be achieved in the case study's marketing department.

Perspectives

For the first time, the marketing department could quickly look at information from various vantages. By varying the dimensions of a situation, new insight could be gained. With the example of the discount structure and profitability analysis, factors as affecting the bottom line could be quickly understood. Factors that had little bearing were quickly identified and ignored.

Flexibility

The ability to change assumption parameters and to re-iterate views of information gave the marketing people the ability to get the right data to the right people in the right format. As an example, one marketer overheard an executive request information about a given product and where it was used. The marketer used Information Access to retrieve the desired information and download it to a PC for graphics input. Through Charting Gallery a pie chart was created addressing the executive's questions - all that took less than 3 minutes. The result: one very surprised and pleased executive!

Responsiveness

As the previous example illustrates, the information center implemented through HP's Information Access provided the marketing department with unparalleled responsiveness.

Opportunity

Because the marketing department could look at the information from new perspectives, new ideas of how to handle situations became apparent. These new views spawned new ideas.

Competition

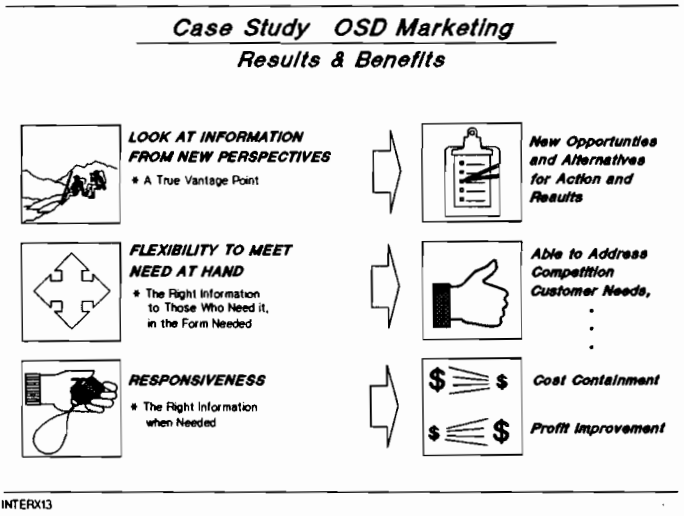
When the competition made certain price and configuration changes, the use of Information Access with Lotus (R) 1-2-3 (R) allowed quick responses. This included the ability to understand impacts on various customer segments.

Cost and Profit

As was noted earlier, pay back for the information center (implemented through Information Access) was quickly achieved. There are other examples A cost

accounting department achieved a pay back in an estimated 3 months compared to the manual methods used and about 1 year faster than custom written programs to implement a information center.

A large banking corporation has cut certain personnel function times by an order of magnitude.



SUMMARY

The marketing department in our example quickly established the utility and practicality of the approach of using the information center through Information Access. The MIS department was able to provide leadership in maintaining standards, responding to needs, and exercising their charter of managing the information asset. What began as a no-win situation turned into a win-win situation.

Information Centers are a natural companion to transactions processing systems. They provide managers and professional staffs with the day-to-day information necessary to remain competitive, to make informed decisions, to monitor quality and to understand what is happening with their businesses.

Information Access gives one the ability to implement these centers in a cost effective, straight forward way, using state-of-the-art methods.

WHERE IS THE METHOD TO PROTOTYPING?

David M. Wattling
EDM Management Systems Inc.
Suite 600, 10015 - 103 Avenue
EDMONTON, Alberta
T5J 0H1

INTRODUCTION

The advent of the so-called "fourth generation" of technology (4GT) has brought with it promises of massive productivity gains for the Data Processing industry. Powerful fourth generation programming languages (4GLs) are claimed to reduce system development costs by factors of ten or more. Furthermore, the use of prototyping techniques with these tools promises to reduce documentation and ensure the right system is built quickly by way of an iterative process.

While these new tools and techniques can certainly reduce system development and maintenance costs, they do not provide the panacea some advocates would have you believe. There are real dangers associated with employing these "power" tools - not the least of which is control. A strong misnomer exists that the use of 4GT permits you to "throw away the book" and therefore all you have learnt about data processing over the years.

Prototyping is sometimes viewed as the method by which to build fourth generation systems. This is false. While 4GLs provide the tools, and prototyping represents one of the techniques you would use in building an information system, a management framework is still required to ensure an organized and controlled project. Without this control, prototyping can become endless, analysis may be cursory and the project scope may be ignored.

This article will begin by exploring what prototyping really is and what it is good for. Then we will look at why a management framework is still required when prototyping a system. Finally we will describe a methodology - the Evolutionary Development Methodology© - that has been designed specifically for fourth generation systems development.

PROTOTYPING

Definition

Prototyping is a technique that is evolving in an attempt to not only reduce system development costs but also to improve the systems ability to meet ongoing user needs. While the technique can be used in many ways depending on the project environment, it always involves the creation of some sort of system model to aid communications between the user and the developer regarding system requirements.

Usage

The traditional way to develop a system was to document, at great length, how a system would perform the functions required by the user, prior to constructing any programs. Inevitably these "specifications" were not understood by the user, or, in some cases, not even read. Hence when the system was implemented, it was quite often criticized as not being what the users wanted.

Prototyping offers a way in which communication can be effectively established between user and developer, through the use of system models. The models become the focus of requirements discussion, as both parties can relate to the models equally well.

There are many ways in which prototyping can be utilized for any given project. A generally accepted definition shows us four levels:

* Mock-up.
* Simulation.
* Working Model.
* Evolutionary.

The first three levels are aimed at better defining the system requirements prior to system construction. They use models that vary in sophistication to provide the system specifications. The Evolutionary level, however, strives to evolve the system into production through a series of progressively more functional models.

Strengths

Prototyping is a very powerful technique for ensuring that the system that is eventually implemented is in reality what the users want. It's primary strength lies in the use of unambiguous models as the communications medium. When users and developers can communicate using a dynamic, working model of the application, most of the problems owing to misunderstanding simply disappear.

Prototyping also fosters a feeling of user involvement in the system development process. They can understand the models and can see that their feedback is incorporated quickly. In essence, they become active members of the project team, and as such will feel ownership of the resultant system.

Weaknesses

Prototyping is not for everyone or every project. As mentioned earlier, there are many ways in which prototyping can be employed. It is important to recognize the type of prototyping required for any given project, ie. should we be looking at the user interface, or is performance the key issue? It is also quite possible that, for major transaction processing systems, the use of prototypes is not appropriate or should be limited to certain user:system interfaces, for example the enquiry screens or informational reports, while the rest of the system is pre-specified as the requirements are given.

Prototyping requires a lot of user time throughout the system development period and it requires developers who are good communicators. It also demands that the users and developers work as a team, with one single common goal - a system that performs the functions required by the users.

Probably the single biggest danger associated with using prototyping, is that of losing control of the project. There have been many horror stories regarding prototyping. Some deal with using a 4GL for the wrong type of processing, but mostly they address some form of loss of control. We hear of "rampant prototyping" where the developers never stop refining the system. We also hear of developers sitting down at the terminal without adequately analyzing the problem first and therefore creating a system without a solidly designed base. Similarly, we see systems start into construction without a full data analysis to provide a firm foundation for the functional development.

In summary, without control, prototyping degenerates to a "seat of the pants" approach to system development - one which the DP industry left many decades ago.

METHODOLOGIES

Definition

A methodology is the management control framework within which system development takes place. It provides the planning and organization for the resources of a project, and defines tasks, sequencing, checkpoints and deliverables.

Need

It is our view that for any project to be successful, three elements must be present - a methodology, a series of techniques and a set of tools. We call this "the tripod theory". In a fourth generation environment, while 4GLs provide the tools, and prototyping provides the primary technique, a methodology is still required.

Use of the fourth generation "power" tools without the appropriate control mechanisms will simply create problems more quickly than with traditional technology. Fourth generation systems can and have been created that were too inefficient, too costly, or were simply inappropriate for the job they were supposed to do.

Options

There are many system development methodologies available. Some are relatively less detailed than others. Some go to a level of detail that includes step-by-step checklists for each activity to be performed by the developer. Many are well developed and have been proven to be effective in a traditional environment (e.g. COBOL, batch, mainframe).

Traditional development methodologies (TDMs) have a number of features in common. They all employ a highly structured and prescriptive approach. They typically concentrate on the beginning of the system life cycle - development to the point of initial implementation. Finally, they incur a large overhead in their requirement to produce highly detailed specifications on paper, which can be up to 50% of the cost of the project. We define these detailed specifications as an overhead, because specifications are not part of the final product - the operational system delivered to the user.

Many of these TDMs have been modified to include the use of prototypes to assist in the specification process. As such they can employ the first three levels of prototyping as defined earlier. However they are not geared to take full advantage of evolutionary prototyping and the use of 4GLs. As an example, it seems inappropriate to create several pages of specifications for a reporting program that may take a dozen lines of code to actually produce.

As such we contend that the traditional methodologies are much less appropriate in a fourth generation environment. In essence they are not geared to give the maximum payback from the use of 4GLs and prototyping. In order to reap the much talked about productivity benefits of this level of technology, the methodology, techniques and tools must all be working in concert.

Rather than trying to force-fit the new technologies into traditional approaches, some methodologies have been created specifically to harness the full power of 4GT. The Evolutionary Development Methodology® is one of those approaches.

EVOLUTIONARY DEVELOPMENT METHODOLOGY•

Philosophy

The Evolutionary Development Methodology• (EDM) is a systems management framework that provides for the controlled use of 4GT, including 4gls and prototyping techniques.

The intent of EDM is to deliver, in as short a time-frame as possible, cost effective, functioning systems that meet real user needs. To this end, business requirements are prioritized and the most important automated first. Remaining business requirements are re-prioritized and the next most important requirements automated - a process that continues throughout the lifetime of the system. As such, EDM incrementally evolves the system to meet ongoing user business needs.

Incidentally, as a result of this process, there is no clear distinction between development and maintenance. Risk is minimized, as each stand-alone increment is not only cost-justified in its own right, but is also relatively small as compared to traditional development.

Emphasis is placed on user involvement from the start, thereby placing responsibility for system requirements definition and time-lines on the user. It keys on demonstrating results quickly - results that the user can relate to. It demands maximum user involvement and commitment throughout the project.

Overview of the Phases

An EDM project has five phases as depicted below:

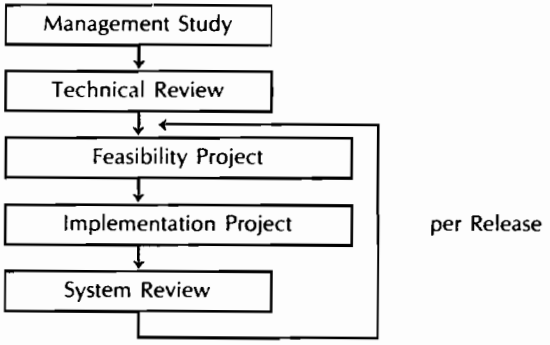


Figure 1. The Phases of EDM

The Management Study and the Technical Review are performed once, at the beginning of the project. The other three phases - the Feasibility Project, the Implementation Project, and the System Review - together make up a single Release. Releases are iterated throughout the lifetime of the system.

The Management Study

This is performed at a very high level. Its purpose is to determine whether or not a computer project exists. The Study would typically be performed before the DP department is called in. One of the purposes of the Study is to avoid heavy investment in automation projects which turn out to be inappropriate or insufficiently cost-beneficial. The Study addresses the current situation in terms of the overall business goals and objectives, and defines how well the business requirements are being satisfied. This includes an analysis of current problems as well as future needs.

Options are then evaluated which will not only include proposed resolutions to the existing problems, but which will also demonstrate themselves capable of meeting both present and future needs. The project scope must be defined at this point. Depth in the definition is not required, but the boundaries must be firmly set. The Study is performed by a person with much experience in the business, preferably by user management itself, or perhaps by an outside consultant.

At the conclusion of the Management Study, two basic questions can be answered, namely: "What action is required to meet current and future business needs?", and "Does an EDP project appear to be the chosen strategy?".

The Technical Review

This phase is initiated by senior user management when the answer to the previous question ("Does an EDP project appear to be the chosen strategy?") is "Yes". This phase takes the business requirements and proposed solution strategy and establishes the technical framework for the solution. It examines and verifies any technical assumptions made during the Management Study.

We need to know the best implementation environment e.g. a mainframe, a mini, or a micro. We need to know communication requirements. We need information on volumes and growth factors. We need to know whether the system will be custom-programmed, or whether some or all of it will be package-based. The purpose of this phase is to answer these questions.

It is possible that more than one (equally) viable technical solution presents itself. This is rare. However, it does happen occasionally, and in this case, each option is explored in more depth. Consideration

is given to both functionality and cost- benefit, but analysis is performed at only a high level. Should the choice still not be obvious, then a Feasibility Project would be performed on each. It should be noted that this is an extremely unusual situation. In nearly all cases, the system size and hence target environment is such that one technical framework is evidently far superior to any other.

A preliminary data analysis and model is mandatory at this stage. It need only be a coarse-grained model, but it must be present. To be able to easily increment the implemented solution, a stable data base is necessary, and this means a validated logical data model of the whole system must be in place.

The Review is performed by a experienced Systems Analyst, who has seen a lot of different sizes and types of applications. Without this experience, the person will not be able to operate on the very high-level data that is all that will be available at this stage. Expertise from other areas, e.g. data modeling, can be brought in as required.

The bottom line from this phase is an answer to the question "Given that an EDP project appears viable, what will the system look like, and what will it cost?".

Action by user management will be either an approval in principle to proceed with development, or a decision to kill the entire project.

The Feasibility Project

This may be initiated by senior user management either after a Technical Review (for Release 1), or after a System Review (for Release n). The primary purpose of the Feasibility Project is to accumulate **just sufficient** information to allow user management to make the decision "Yes - this Release is a go" or "No - this Release is not worthwhile - forget it". It is important to remember that the Feasibility Project affects the fate of just that one Release, and not the entire system. Obviously, if Release 1 - the most important - is not feasible, then serious doubt must be cast on the project as a whole. But if, for example, Release 17 proves infeasible for whatever reason, then that does not cast a shadow over Releases 18 through 24.

Each Release, while it may depend on the implementation of prior Releases, is nevertheless functionally independent. It must also be analyzed in its own right with respect to costs and benefits. To do this, both the overall scope and the detailed functional definition of the Release must be defined precisely, as well as a strategy for implementation.

In support of the functional definition is the data model. This is how we maintain coherence through Releases. The data model is evolved along with the system. In particular, each functional Release includes

a refinement from a coarse-grained model to a fine-grained one for that particular area. In the Feasibility Project, just enough refinement work is done to be able to support the basic purpose of the Project; full refinement will be done in the next phase - the Implementation Project.

The cost-benefits of this Release must be analyzed in detail. Since we are dealing with an evolving system, the integrated and projected cost-benefit of the system as a whole is a key factor in the go/nogo decision. The numbers resulting from this exercise should be compared with and verified against the Release Strategy, both to validate the Release Strategy, and to act as a check on this individual Release.

A skeleton prototype of the Release would normally be constructed. This serves three purposes. Firstly, it will prove (or disprove) that the Release is technically viable - although this is not the only factor in the go/nogo decision. Secondly, it will act as a vehicle for the system users to validate scope and requirements. And thirdly, should the decision be taken to go, then the prototype will act as the basis for evolution into implementation.

The bottom line from this phase is an answer to the question "Is this individual Release a go, or not?".

The Implementation Project

This is initiated by senior user management following a "go" decision at the conclusion of the Feasibility Project. The purpose of the Implementation Project is to deliver a working information system that meets the business requirements as defined in the Feasibility Project. This will include software that is validated by the user, produced in an evolutionary fashion with active user involvement and implemented as a maintainable entity, ie. with documentation, converted data and installation.

The Implementation Strategy as defined in the Feasibility Project is revisited and refined. Typically, the first task performed in the Implementation Project is to finalize the data model for the subject area for this specific Release. The Implementation Strategy also includes a proposed list of Stages for the Implementation Project. Each Stage is a small, manageable "chunk" of work, which when taken together make up the business function(s) the Release is designed to automate.

Following the finalization of the data model and implementation into a physical database structure, we move into the Stages. This is the heart of an EDM project. Each Stage is a meaningful piece of work in its own right (for example, a customer file maintenance subsystem). All Stages for a Release are needed in order to fulfil the business function. Each Stage is built and validated as a separate sub-project within the Implementation Project. Acceptance testing proceeds in

parallel with Stage construction. Typically, the end user will work with the developer during construction, to ensure requirements are fully met, and will also test the Stage independently following completion of construction. A Stage includes software, documentation, user manuals, and whatever else is necessary to provide for its functionality. It is a true deliverable.

Each Stage is normally prototyped, although there can be variations. What this means in practice is that the developer will create a first-cut at the software, probably based on the Feasibility Project prototype, and aim to hit at about 70% of user requirements. This is a key number, and crucial to the process. Based on user feedback, the developer will refine the software, adding value to the extent of 70% of the remaining requirements each time. After five (very fast) iterations, some 99.7% of requirements should be met.

The bottom line from this phase is "Delivery of an implemented and accepted business function".

The System Review

A Release is not completed until the System Review is done. This phase of the Release looks simultaneously backward to the work that was just completed, and forward to review priorities, budgets, and future Releases. This phase is most critical after Release 1 - this is the first opportunity to perform substantive quality assurance on the Management Study and Technical Review.

The first function of the System Review is to review the success of the Release. Specific points that should be covered include a check of delivered versus planned functionality, and actual costs and timeframes versus forecasted. For Release 1, it is important also to review the actual development process, answering such questions as "Was there indeed good, close user:developer communications?" and "Was strong management and control actually exercised by the user during the project?".

The second function of this phase is to look to the future. In addition to the reviews mentioned above, the System Review must also answer the question "How does what was learnt during the last Release affect future Releases?". Remember, the intent of EDM is to incrementally evolve a system that will meet ongoing user business needs. The System Review serves the critical function of keeping the overall strategic direction on target.

The bottom line is "user management informed and in control".

CONCLUSION

Control

EDM recognizes project control as a critical factor in successful systems delivery. Control has been built in as an **integral** part of the methodology. Some of the inherent control factors are:

* Multiple checkpoints are based on meaningful deliverables.

* Extensive user involvement is mandated.

* Minimized risk due to the Release by Release approvals and delivery, and therefore the shortest possible time to working software.

* Quality assurance is built in, through the integral nature of model refinement and testing.

* Change management is inherent.

Summary

EDM represents a new concept in system development approaches that is driven by the users and their business needs. It provides strong management control over the use of 4GT so as to minimize the risk of using these "power" tools. It is the final element required to augment fourth generation tools and techniques, thereby facilitating successful fourth generation systems development.

The combination of EDM, prototyping and 4GLs represents a powerful approach to system development. One that can realize the massive productivity improvements promised by this level of technology, without sacrificing management control.

EDM changes the emphasis of the system development success criteria from the traditional "On time, under budget, to specification" to **"Business needs met, on an ongoing basis, within budget limitations"**.

The bottom line is that the effective use of any advanced technology is not just the employment of a new programming language or a new technique, but it is a new way of conducting DP business.

ACKNOWLEDGEMENT

Evolutionary Development Methodology® is a trademark of, and identifies a proprietary product owned by, EDM Management Systems Inc.

Disaster Recovery — Beyond Planning
Debrah A. Whitaker
Albuquerque Publishing Company
7777 Jefferson NE
Albuquerque, New Mexico 87109

Disaster Recovery: Beyond Planning

Presentation to Interex North American 3000 User Conference
September 20-25, 1987

Table of Contents

5. Contingency Site Description

- 5.1 Location
- 5.2 Nature and Terms of Agreement
- 5.3 Configuration
- 5.4 Accomodations for Staff/Control Center Operations
- 5.5 Scheduling
- 5.6 Procedures/Documentation

6. Recovery Procedures

- 6.1 Team Assignments
- 6.2 Implementation of Procedures
- 6.3 Activation of Control Center
- 6.4 Feedback Loop with Top Management and Users
- 6.5 Activation of Off-site Storage and Contingency Sites
- 6.6 Implementation of Replacement Data Center
- 6.7 Phase-in of Return to Normal Operations

7. Testing and Ongoing Maintenance

- 7.1 Test Plan
- 7.2 Policies and Procedures
- 7.3 Review and Update

Conclusion

Introduction

Numerous MIS publications that have crossed our desks recently contain at least one article on the subject of Disaster Recovery and Disaster Planning. These terms have taken over the news slots that were once devoted to Data Bases, Personal Computers and, more recently, local area networks. As we all know, the articles written on each of these subjects basically tout, "I've never seen one, but I'd know it if I saw it." These products are now known as "vapor ware" because the products themselves are publicized, and even sold, but do not really exist.

Such is the case with Disaster Planning and Disaster Recovery. The statistics tell us that "possibly less than 20%"[1] of the big shops subscribe to some type of recovery services and only 15%[2] of all the nations' companies have plans in place. A recent Supergroup Association article points out that the disaster recovery plan most often used by MIS managers is an updated resume.

Is Disaster Planning really as important as the consultants tell us? It is when you consider that "70%" of "all businesses that sustain" a significant interruption " of services due to some sort of disaster . . . "go under."[3] Best case, CEOs may face a strongly worded recommendation from auditors within the management letter for fiscal year 1987 and an audit opinion statement in 1988 if the plan has not been developed and tested. Auditors disagree among themselves as to when Disaster Recovery will become a regular audit point. They do not seem to disagree that it is merely a matter of time.

It is not the intent of this paper to list the various horror stories explaining what happened to those companies who had no means of recovery. It is also not the intent of this paper to focus entirely on how important and vital the planning process really is.

The primary purpose of this paper is to develop a framework for developing and integrating a Recovery Plan into the current ongoing activities of the MIS department and the company as a whole.

We have focused our comments around four primary concepts.

1. No disaster is ever planned.
2. There are disasters other than floods, fires and earthquakes.
3. Planning is done not only to prepare for disasters but to prevent disasters.
4. If the disaster doesn't get you then the auditors will.

Throughout this paper the terms "Recovery Site" and "Contingency Site" will be interchangeable. We will begin this paper by defining Disaster Planning and Disaster Recovery so that the remainder of the paper is in the proper perspective.

[1] Datamation, Feb. 1, 1987. "Disaster Recovery: Who's Worried?", David Stamps, page 60.

[2] Uptime Newsletter, June/July 1986, page 1

[3] IBID

For the sake of this paper, we will define a "Disaster Plan" as "a document prepared by MIS personnel in conjunction with top management and system users for the purpose of documenting and developing procedures to handle every conceivable occurence which would result in the destruction or inaccessability of vital company records for an unacceptable length of time." These records may be manual or automated. The assignment of some measurement of "acceptable" or "unacceptable" length of time must be defined within the context of the planning document. Some businesses, by the nature of their product or service, may be able to function for a substantial period of time without access to critical data or records. Other businesses may suffer great losses if data is not immediately recovered.

Although a checklist should initially be used to create this document, the document itself should be a tailored set of tested procedures, instructions and documentation for handling these emergency situations.

The Recovery Plan should be a living document. In other words, after initial preparation, the plan should be updated as goals change, equipment is replaced and software modified. If the data center for which the Recovery Plan is being prepared is not standardized or well organized, a full recovery plan will be difficult, if not impossible, to prepare. However, if solid procedures are in place, the Recovery Planning process will have the affect of fine-tuning procedures or standards.

There are a number of levels of recovery which must be addressed within the context of the planning process. In order of significance, these levels include:

1. Primary hardware (mainframe, tapes, disks, etc.)
2. Primary applications (in order of priority)
3. Critical Data
4. Users
5. Secondary hardware (distributed mainframes, PCs, OSI, etc.)
6. Secondary applications and data
7. Casual users
8. Manual records
9. Telephone and mail communications

This not only allows for an orderly template from which the plan can be created but also allows the plan to be chunked into logical pieces. Completion of more critical portions of the plan can thus be completed before less critical portions.

"Disaster Recovery" will be defined as "the process of successfully implementing the procedures and instructions documented in the Recovery Plan in a minimal amount of time that is the least disruptive to ongoing business operations with an ultimate goal of returning to normal operations.

Section 1 — The Readiness Team

1.1 Purpose

The primary purpose in defining and assigning a group of people as a "Readiness Team" is to provide some clear direction to people at a time which will, most likely, be otherwise confusing. If people clearly understand their responsibilities and what is expected of them, some of the uncertainty is mitigated. If team members have been through a "dry run" of an emergency situation they will have a greater understanding of procedures, and careless errors are less likely to occur. Therefore, the purpose of a Readiness Team is to provide a qualified group of trained people capable of handling and assisting in recovering data in a crisis situation.

1.2 Using Available Resources

In analyzing the available resources, several key issues must be considered. First, the Readiness Team should consist of representatives throughout the organization, not only data processing staff. Team members should include:

- Users of every major system
- Technical Services personnel
- Representatives from executive management
- Operations staff
- Programmers/analysts
- Vendors or consultants
- Auditors

This list may seem a bit overwhelming at first glance, however, these people are currently involved in developing, implementing or utilizing systems at facilities every day.

The second item which must be carefully examined is the composition of the Readiness Team. People who are assigned major roles on the team must be people who work well under pressure. Due to the stressful nature of handling somewhat non-routine procedures at a separate site with new personnel, mistakes are more likely to occur. These mistakes could turn the entire recovery process into a further disaster.

The third element of integrating a Readiness Team is thoroughness. The immediate goal, after a disaster takes place, is to attempt to obtain an "acceptable" level of processing for the critical and strategic systems of the company. The ultimate goal is to eventually return all systems (automated and manual) back to normal operations. Therefore, personnel assigned to the Readiness Team should be knowledgeable individuals who are able to work from a set of detailed task plans.

Keep in mind that recovery to an "acceptable level of processing" is a different set of tasks from full recovery to normal operations. Therefore a different mix of people will be assigned to these tasks.

The purpose in identifying and training a Readiness Team is to focus a specialized group of experienced people on the task of maintaining an acceptable operational status until such time that conversion back to normal operations is feasible. A Readiness Team is managed in much the same way as a software development

team. Larger tasks are "decoupled" to form more manageable sub-tasks or clearly defined steps. People are assigned to these smaller units of work by skill-level and experience.

1.3 Task Assignments

Task assignments should be given to personnel based on strengths, rather than position or interest. The recovery period is not the time to learn new procedures. Likewise, staff members will have little spare time to teach others new tasks or new jobs. Task assignments should also correlate as much as possible to ongoing job responsibilities. Analysts should not suddenly take on Data Base Manager tasks nor should MIS directors be "quick-fixing" errors in COBOL code during the recovery period. This is not to say that team members will not be required to perform more tasks and work additional hours. The primary point is that task assignments should be made in close alignment to ongoing job functions and strengths.

1.4 Team Coordinators

The assigned Readiness Team Manager must be a decision maker who has the authority to act quickly on large purchases as well as one who can have total access to the system until full recovery is achieved.

Although it is hoped that the recovery plan will cover as many situations as possible in as much detail as possible, no plan will cover every aspect of a disaster that might occur. Therefore, the Team Manager must devote a great deal of time fine-tuning or interpreting the plan. This leaves little time to manage individual team members.

Usually the best way to manage the team is to assign Team Coordinators to be responsible for specialized areas or tasks. These coordinators must manage other team members and report to the Team Manager on progress or problems. Although it is important to add as much structure as possible to the Team so there are few uncertainties, it is equally important not to get carried away with multiple layers of team members. "Red Tape", or bureaucracy, has no place during a disaster. Team coordinators and team members need access to each other as well as to the Team Manager.

1.5 Alternate Team Members

Disasters rarely occur when people are totally prepared. Although MIS Directors can put people on-call in case problems arise, they cannot determine a vacation schedule based on when a disaster is most likely to occur. Therefore, absences due to vacation or illness, causing personnel to be unreachable, must be anticipated.

The best way to plan for these absences is to assume that no one assigned to the original Readiness Team will be available when a disaster occurs. This assumption will require two primary action items:

- assignment of alternate team members
- cross-training of personnel

Key members of the Readiness Team should be assigned true alternate personnel

-- or personnel who do not currently have assignments as members of the Readiness Team. If this is not possible, replacements should be assigned from the general team members. Team Coordinators should not be replaced with other Coordinators. Consultants who are familiar with the site may be utilized if staffing is unavailable.

The reality of assigning alternate personnel stresses an important point. Difficult choices must be made in order to make logical team assignments and to "back-up" each assignment with alternative personnel. By definition, the original Readiness Team cannot be composed of your entire MIS or Technical Services staff. Alternate team members may be used simply to relieve staff who have been working extended hours.

In any case, assignment of alternate Readiness Team members requires a focus on cross-training of individual team members. Although cross-training is desirable in most shops, many MIS departments continue to assign one analyst to each specialized application. If a disaster strikes and one analyst is ill, or vacationing in Hawaii, an already difficult and stressful situation will become unmanageable and ultimately unrecoverable.

Section 2 — Major Considerations

2.1 Applications

One of the exercises performed at most disaster recovery seminars or presentations these days is requesting each participant to prioritize applications in numerical order 1-n. This exercise is also suggested as a first step in most disaster recovery kits or checklists available on the market today. The primary point that needs to be made in performing this exercise is that during a disaster there will be contention for access to vital records. It must, therefore, be determined, within the parameters of a plan, which records are the most vital and required to keep the company running. This prioritization is not as simple as it appears. These priorities should be worked out with top management. Although MIS personnel may have a good grasp on the importance or magnitude of various systems, priority setting is a strategic process that should involve the Board of Directors, the CEO, the Strategic Planning Committee and other decision-making bodies to determine, not only which records are the most vital, but also which records will be the most vital in one or two years. These priorities will vary from industry to industry and from company to company.

Although we are not suggesting the user community be omitted from the prioritization, it should be mentioned that end users will most often place their individual applications at the top of the list of priorities. The process of developing and maintaining a plan is extremely time-consuming. Top management should have the major input on deciding priorities.

2.2 Scheduling

The entire scheduling issue remains an important factor in the day-to-day operations of all computer centers. Scheduling implies routine procedures and processes (hopefully documented) for people, jobs or maintenance. Once again, if clear concise procedures are developed for all known situations, recovery from unpredictable or the "unknown" will be less stressful, more controllable and, therefore, easier.

The base-line for these procedures should be the current schedule (e.g. when payroll normally runs, when back-ups are taken, etc.). These schedules should be maintained off-site with other documentation.

The schedule during the Recovery Period will, no doubt, be somewhat different due to the fact that the short term goals of the Recovery Period will, in all probability, be different than normal operations. An environment which had previously concentrated on identifying and developing new strategies and new systems will be tasked with monitoring and maintaining. Schedules and scheduling procedures should reflect these alternative functions. The time once spent on new development efforts will be spent on getting systems back to normal operations or, at minimum, to achieve the same level of performance provided prior to the disaster.

2.3 Users & Organization

Throughout this paper we have hinted at the tasks which should or should not be assigned to the user community. The primary purpose of this section is to

focus more clearly on defining the scope and the relevance of user participation in Disaster Planning and Disaster Recovery. As may already be known, users are probably the most valuable resource we, as analysts, have in identifying, defining, designing, developing, and maintaining information systems. We have been utilizing key users as project team members for a number of years. The user community has, as a result, become more sophisticated, less timid and, rightfully, more demanding of systems' professionals. Users in many companies have a full functional knowledge of the systems they work with on a regular basis. Many users are "walking data-dictionaries" of every field definition, report format and scheduling process. As such, users have become a valuable part of each system developed or maintained.

Users, therefore, must actively participate in development and implementation of a Recovery Plan. Although it may not be a good idea to involve all users in the company-wide prioritization process, users know what data is vital to their departments versus what data may be defined as useful. The user community should be able to assist Readiness Team members in identifying non-essential jobs or reports which can greatly relieve recovery operations until full recovery is achieved.

Users should be given specific task assignments on the Readiness Team. A primary task to which users can be assigned is that of liaison with their appropriate department. During the recovery period, users can accurately relate progress to department personnel and provide effective "P.R." in efforts to attain an acceptable status and eventually return to full operations.

An effective way to communicate responsibilities and job requirements to users, technical staff and MIS staff is to jointly develop job descriptions and an organization chart which clearly define relationships among the Readiness Team members. These job descriptions and the organization chart should be incorporated into the recovery plan.

2.4 Training

Training will take on a different dimension during a recovery period. Training material, documentation, slides or other training aides should be stored off-site with tapes and disks. This material will be vital if equipment or software must eventually be replaced. Training material can also be a vital resource if for any reason key members of the Readiness Team are not available for the recovery period. Informal, quickly scheduled training sessions may be required if team members are not fully knowledgeable on every aspect of assigned tasks. The Planning process should be used as an opportunity to evaluate the effectiveness of a site's training material.

2.5 Equipment

Many times a disaster is a direct result of hardware, network or other equipment failures. Although equipment can be repaired, it is quite often cost effective to replace the equipment. If replacement is determined to be the best alternative, much attention should be focused on identifying the reason for the original failure and the approach to replacement. Faulty equipment should not be replaced with other faulty equipment. Research must be done in order to determine if the cause of the failure is inherent to the equipment or the configuration of the equipment. If the research proves to the team's satisfaction that duplicate

replacement would not solve the identified problems, the off-site recovery period may need to be extended long enough to initiate and complete a hardware search process. In addition, software may require modification in order to function properly on new equipment with a different configuration.

This extension of recovery time should be planned for and discussed with hardware vendors or recovery service bureaus in a "what if" mode. Procedures for this type of situation should be developed and incorporated into the company's Recovery Plan. This portion of the plan should be updated as new hardware and software is purchased by your organization. Alternative hardware configurations should be identified within the plan.

2.6 Location of Back Up Equipment

A primary consideration of the Recovery Plan is the location of the back-up equipment. It is obviously easier and more convenient to test the equipment or implement the plan if the recovery site location is near your facility. There is also some risk involved in identifying a location which is too close. If your area is prone to such natural disasters such as floods, earthquakes or tornadoes, you may want to ensure the backup equipment location is not in the same area as your facility. Some service bureaus offer equipment backup facilities on trucks which may be dispatched from a central location to a site of your choosing when a disaster hits. Some consideration should be given to the fact that trucks or vans are known to break down, sufficient power is necessary to run the equipment and these traveling vans could be tied up at other sites.

If a remote contingency site is chosen, the Readiness Team must be flown to the back-up location which may be a substantial distance away from your location. More discussion of the back-up site is provided in Section 5 of this paper.

2.7 Manual Back-Up

Although the majority of this paper is dedicated to providing alternatives for backing up automated systems with other automated systems, manual back-up procedures should not be overlooked. Each system should be examined closely for the feasibility of providing manual processing during any extended down time period. Source documents may be filed and kept for after-the-fact entry into an automated system. The systems which typically lend themselves to manual back-up are systems such as general ledger, accounts payable or accounts receivable systems. For the most part, the paper source documents for each of these systems are currently filed and tracked for audit purposes. Although the volume may prohibit the use of manual back-up, the possibility should be thoroughly explored on a system by system basis. Microfiche or microfilm should be considered for high volume systems. Manual back-up procedures should be incorporated into the Recovery Planning Document.

Section 3 - General Procedures

3.1 Fire

Although even a small fire or the accompanying smoke can cause tragic results to critical data or equipment, the key to recovery is preparedness. Smoke and fire alarms should be located in and around the computer room where the equipment is maintained. Operators should be trained in evacuation routes and in the use of fire extinguishing equipment.

Sprinkler systems are installed today in most buildings. If, however, the resources are available, halon systems should be installed. Halon systems are designed to extinguish fires without damaging equipment or people. Often sprinkler systems cause as much, or more, damage to equipment as fire or smoke. However, Halon systems without accompanying sprinklers do not always meet standard fire codes. Some discussion with fire inspectors and insurance companies must take place if halon systems are installed in place of sprinkler systems. The operators should be trained in how to control any sprinkler system as well as how to extinguish a fire. Some consideration should be given to configure hardware in such a manner that critical hardware is not directly beneath sprinklers.

If your computer center does not operate on a 24-hour per day, operator-present schedule, it may be wise to develop a system which automatically calls security, the operations manager and/or the fire department should a fire occur. This type of communication may be set up through your current security system utilizing your telephone system.

Back-up tapes and disks should be maintained both on-site, in a fire-proof safe, and off-site to ensure the possibility of full recovery. If storage media and equipment are non-functional or are completely destroyed, the off-site Recovery Plan should be implemented.

3.2 Power Outages

The majority of power outages occur for a matter of seconds or minutes. Some preventive measures should be taken to ensure that these brief outages or surges will not damage any processing that is taking place or affect critical equipment. All power in the equipment room or tied to any terminal should be "clean" power. Communications between or among systems utilizing networks should use shielded cable. Surge protectors should be placed on all stand-alone equipment or equipment such as PC's. Surge protectors will, however, be inadequate for any power outages lasting longer than a few seconds. If power is not available for 3-15 minutes, valuable processing time can be halted and data may be lost. Protection may be accomplished through the use of a battery back-up power source or an uninterrupted power supply (UPS). Most of these power supplies offer a 10-15 minute window. Since power outages are usually unpredictable, the 10-15 minute window provided should be used to alert and sign off all users, complete or abort any jobs and accomplish normal shut down of the disk units.

tape units and processors. If time permits, file statistics should be produced for later comparison. Power may be restored immediately, however a conservative approach will be the safer approach. Clear, concise procedures for power outages should be developed and incorporated into both the disaster plan and into an operations procedures manual maintained in the computer room.

If a power outage occurs and processing is abnormally terminated, resulting in loss of data, unrecognizable application errors or operating system errors the system should be thoroughly tested before processing resumes. Full diagnostics should be performed to determine whether or not equipment was damaged. If maintenance contracts are in place, vendors should be contacted immediately. Diagnostics should be performed in an attempt to determine where or when data may have been lost. This analysis can be performed only if analysts or the Data Base Manager are monitoring file and data statistics on an ongoing basis.

If it is determined that the system cannot be properly restored after a power outage due to software problems, back-up copies of the operating system or applications software should be reinstalled. Data back-up files can then be reinstalled. Note that these back-up files should be copied prior to reinstallation in case data or applications are destroyed accidently during the reinstallation process. These extra back-up copies could be produced at your recovery or reciprocal site. Assistance should also be requested from your local vendor if necessary.

3.3 Communications Failure

If your data center supports remote processing via networks, leased lines or modems, sufficient planning must be performed in order to ensure that:

- Communications problems do not disrupt your entire business activity.
- If primary communications lines are unavailable, other communications methods may be used.
- Any chosen recovery site will be able to accomodate communications requirements.

Communications failure must be sufficiently addressed in the Recovery Planning Document. However, this portion of the recovery plan is highly dependent on the type of communications equipment in place at your site, the configuration of the communications devices and, most importantly, the scope of the data dependency. Not all of the possible combinations will be covered within the bounds of this paper, however many of the more common ones will be discussed.

First of all, we will place communications equipment into three category types:

- Modems
- Direct Linkage
- Network Linkage

Modems can be backed up by maintaining "spares". Spare modems should be compatible and if primary modems are security-dependent, then back-up modems should also be security-dependent. As we all know, modems are usually used in conjunction with telephone lines. Even the most reliable modems are only as good as the telephone line utilized at the moment of transmission. Telephone lines were not originally designed to pass every data format. Voice communication is much more forgiving than digital data communications and therefore not required to be as reliable. If your data center sends or receives data over telephone lines, some precautions should be taken to ensure the data is accurate and to protect from potential disaster. If data is being sent via modems through remote terminals, special editing routines may be set up to check for erroneous data, or "noise", being sent prior to system up-date. There are a number of design techniques available to accomplish this process depending on how the data is being sent or accepted. If telephone lines are down, back-up arrangements should be made to use alternative lines or express-mail source documents to a nearby site if available or to the central computer site if necessary.

Direct linkage is normally utilized for communications within a facility or complex. In a few cases, direct linkage is used outside the immediate area but such communications can quickly become cost prohibitive and, therefore, difficult to "back-up". However, some precautionary measures can be taken to ensure that some level of communications capability is available. If, for example, network lines are being utilized, these lines should be shielded to prevent "noise pollution" from other lines. Modems can be used as a back-up alternative if communications lines are unavailable. The systems must, therefore, be configured to accept either direct communication or data received via modem. Technical services staff members, communications specialists or vendors should be contacted immediately when problems are identified to ensure that repairs are made in a timely manner.

Leased lines are normally utilized for long distance communication purposes or for wide area networks. The majority of leased lines are dedicated telephone lines and the same precautions, outlined in our discussion of direct linkage and telephone lines, should be employed with leased lines. Leased lines are usually more reliable than standard telephone lines because they are dedicated, or used only to transport data, however even these dedicated lines are subject to failure from weather, human or natural disaster conditions.

Back-up procedures for overnight delivery of source documents to a nearby site or to the central site for data entry or re-entry should be developed. These procedures should be used only if the leased line problems can be resolved in a reasonable period of time. Longer term plans should be developed to move people and source documents to a nearby site, or to the central site when necessary, if these lines are unavailable for long periods of time. Once again, modems should be used as a back-up option in those cases. It is important to note that a full test of the recovery plan should test these various communications options.

As discussed earlier, based on the configuration of your communications devices,

other precautions may be implemented. For example, if your network follows the template of a star configuration, you may want to set up communications between sites in case one leg of the star is inactive. If the network is configured in a token ring configuration, some provision should be made to ensure that each node is fully backed up or may be bypassed. This requirement is necessary in order to ensure that if one node is down the entire network is not down. Whatever approach your site takes, it should be emphasized that any network or communications configuration should be implemented in such a way that pieces of the network may be down without disrupting the entire network. Any back-up precautions taken with CPUs, disk drives, or data will be irrevelent if the entire network is inaccessible.

All of these issues should be addressed in deciding whether or not your site should operate in a highly centralized or a distributed processing mode. There are advantages and disadvantages to either choice. With the proper configuration, distributed processing sites can back each other up. However, it would most likely be cost prohibitive to maintain back-up sites for each distributed location. All of these decisions must be fully documented in the Recovery Plan. Procedures should be included to handle each alternative, site by site, where applicable.

3.4 Flood

Flood or water damage is one of the most difficult situations for which to plan. Of course much of the planning for this type of disaster must be done prior to constructing the building in which the data center is to reside. As a practical matter, data centers should not reside in flood plains. However, cost or convenience usually dictates where a company and its data center will be located. If the recovery planners do not have the luxury of choosing the location of the data center, some additional precautions can be taken in order to prevent damage from flood. These precautions are discussed further in this section.

If the building is located in an area classified as a flood zone, every effort should be made to place the data center on an upper story. If the data center is located at ground level, raised flooring should be installed. Many data centers are designed with lowered flooring for the convience of moving equipment in or out of the computer room. Although raised flooring is somewhat less convenient, ramps or lifts can be created to allow for movement of equipment.

The raised floor will decrease the risk of damaging equipment although cables and power lines may be damaged. If the data center is located in a flood zone, primary wiring should be located in the ceiling and walls as opposed to the floor in order to minimize damage to cables and communication lines.

3.5 Hardware Failure

Although most of the disaster recovery "kits" or checklists marketed today by consulting firms focus primarily on "natural" disasters, the majority of data center disasters may be directly linked to hardware or software failure. In a 1986 survey

performed by the author while working for a Big Eight accounting firm, of 150 randomly selected MIS departments polled, 95% stated that the majority of "potential disasters" and "real down-time" could be attributed to hardware or software failure such as a head crash; not natural disasters such as flood, fire, etc. A typical scenario is:

- 7:00 a.m. - (First day) operator cannot bring system up for normal processing
- 7:30 a.m. - Operations Manager notified after numerous attempts at re-booting the system
- 8:00 a.m. - Operations Manager informs data center director that problem exists, and system will be down for several hours
- 8:15 a.m. - User community and top management notified
- 8:30-Noon - Numerous attempts at bringing the system up have failed. Diagnostics cannot be performed
- 1:00 p.m. - Hardware vendor contacted
- 2:00 p.m. - Vendor arrives on site to diagnose problem
- 3:00 p.m.- Problem discovered
- 3:30 p.m. - Part ordered via phone for express mail
- 4:00 p.m. - Users and Top Management informed of delays
- 4:15 p.m. - Data center staff dismissed for remainder of day
- 11:00 a.m. - (Second Day) - Part arrives
- 11:30 a.m. - Part installed, diagnostics performed by vendor
- 1:00 p.m. - System is brought up and becomes operational for user community

This scenario happens frequently and is highly dependent on capability of operations staff and the vendor dispatched to diagnose the problem, the availability of parts and the willingness of the manufacturer to ship the available part immediately. These issues could increase "down-time" exponentially.

A variety of preventative measures should be put in place in an attempt to lessen the risk of potential hardware disasters. Under ideal conditions duplicate equipment or, for lack of a better term, a mirror-image configuration should be available. If the primary system fails, the back-up or mirror-image system can be utilized. The use of this secondary system will be mostly tranparent to users and will keep both users and analysts productive until the primary hardware can be repaired.

This paper is not necessarily advocating keeping a complete system in cold storage "just-in-case." The author does, however, believe that two smaller systems, each capable of handling the company's critical applications, are safer than a single large system.

Additionally, whenever possible, spare parts should be maintained on-site, or at minimum an agreement should be made with the local hardware vendor to maintain a minimum number of critical equipment parts. Diagnostic equipment

should be purchased and maintained on site. Diagnostics should be performed on a regular basis and monitored for patterns and variations in pattern. Predictive support monitoring by vendors, via phone lines and modems, should be utilized to its fullest extent in an attempt to identify potential problem areas before computer resources are affected.

The primary focus on the subject of hardware failure has been on preventing failure or preventing significant loss of processing capability. All of these items should be considered in the context of a disaster recovery plan.

In addition, planning must be done for those situations where back-up hardware is unavailable or the back-up system also fails. In these cases, the Readiness Team should be activated and the recovery plan initiated. If the hardware is repaired prior to full implementation at the recovery site, time and energy will not have been totally wasted. These situations can be used as a test of the recovery plan.

3.6 Software Failure

Unless a site is installing hardware and software for the first time, there is almost no conceivable situation in which a software failure cannot be adequately planned for and avoided. These procedures should be imbedded in normal operations of any data center. The procedures should include policies and procedures concerning:

- Creating and maintaining a development environment separate from the production environment
- Creating and maintaining a test environment separate from the development or productions environment
- Creating back-up copies of all current applications for on-site and off-site storage
- Creating back-up copies of new releases prior to installation
- Creating nightly back-ups of database files (or at a minimum, changes made to the database)
- Creating and maintaining a back-up copy of the operating system on a separate disk as changes are made
- Creating and maintaining a weekly back-up copy of the operating system
- Creating back-up copies of all package software prior to installation and eventual conversion
- Fully testing any package software prior to installation
- Testing new software releases before conversion to production
- Maintaining primary back-up copies of applications and data in a fire-proof vault
- Maintaining secondary back-up copies of data and software applications off-site in a controlled environment
- Full testing of back-up applications at the recovery or "hot site"
- Operating replacement systems in parallel with current systems to

ensure successful conversion

- Designing structured, modularized software that can be easily maintained
- Training and cross-training analysts on applications

The primary focus for preventing software failure should be on testing, creating back-up copies, and creating back-up copies of the back-up copies. Unfortunately, software can never realistically be completely tested. There is always a chance that an undetected "bug" will make an appearance in the middle of running a critical application while the knowledgeable analyst is vacationing in Jamaica. A well documented set of procedures should be developed in anticipation of such a potential disaster so the situation may be adequately handled and the software repaired. Proper cross-training should also assist in lessening the risk involved in such a situation.

Test data utilized during initial implementation should be kept and maintained as the applications are modified. This test data can be utilized at a recovery or back-up site to determine whether the software failure is due to a hardware problem. This test data would also be required if activation of the recovery site is necessary and again for testing of the replacement site. These issues are discussed in greater detail in Sections 5 and 6.

If a single critical application is not operational within a predetermined "acceptable" period of time, disaster recovery procedures should be implemented for the application in question. Readiness Team members should be contacted and organized. Software vendors or outside consultants should be utilized, where applicable in efforts to get back on-line.

Back-up software and copies of data files should be installed at the recovery site. Base-line test data should be run through the application and results verified. If test data results are accurate, live data should be brought into production at the recovery site.

It is important to re-emphasize that the Disaster Recovery Plan should address all systems, in order, or priority, application by application. No matter how well other non-critical applications are working, unless the most critical applications are functioning properly, critical data is not available and the data center has failed.

3.7 People Failure

Although not often discussed within the context of a Disaster Recovery Plan, people failure can often result in, or cause, most of the other failures discussed in this paper.

Analysts and programmers make errors, go on vacations, quit or become ill. Users become irate. The results may be errors in data entry or coding which do not manifest themselves until knowledgeable people are unreachable. These results could be disasterous.

The primary preventative measures for lessening the risk of "people failure" include:

- Cross training of key personnel assigned to various applications
- Assigning more than one key person to each design or programming job
- Developing a detailed set of procedures and documentation for each application
- Ensuring that design and code "walk-throughs" are performed on a regular basis as systems are developed and modified

The primary key to preventing people failure is spreading the knowledge around. It is important to note that this distribution of knowledge should be applied to the user community as well as MIS personnel. An additional preventative measure involves properly implementing security. Security is discussed in more detail in Section 4.

Section 4 Reduction of Risk

4.1 Protection of Data

As discussed in the introduction and again in Section 3 of this paper, a prime element in disaster planning is anticipation of events in order to avoid possible disasters. The primary method used to avoid disasters is normally through reducing risk. The best way to reduce risk is to concentrate on protecting critical applications and, above all, critical data. A number of suggestions for protecting critical data have been identified throughout this paper however more thorough analysis of these suggestions in this section.

Quantitative risk analysis focuses on the exposure of each system and on the implementation of controls to produce loss expectancy over time. Automated risk analysis software is available which assists in evaluating these controls in order to identify systems vulnerability.

First of all, critical data and critical applications should be defined and prioritized within the content of the disaster plan. Current data dictionaries should be maintained and data definitions should be provided. Data base schemas should be developed and modified as data is restructured or relationships among data changes. Copies of these documents should be stored off-site in a controlled, secure area.

Access to data files must be controlled in order to prevent circumstances in which a disaster is likely to occur. Security codes should be assigned based on "need-to-know" and all access levels should be closely monitored. New sign-on codes should be developed periodically. User IDs should not be printed as identifiers on listings or print-outs for everyone to see.

These procedures are probably standard at most sites, however often the secondary reasons for such security measures are emphasized. Although it is necessary to keep unauthorized people from even inquiring into vital records, such as financials, it is even more important to protect all critical data from being destroyed - knowingly or by accident. The author has observed that much (if not most) of the critical data destroyed is due to authorized users making errors during an update or delete function or to analysts restructuring or restoring files incorrectly with no back-up files.

Fault tolerant software and hardware is available on the market today. The purpose of these systems is to prevent data loss even after a disk crash. These systems actually mirror all data from the primary drive onto a secondary drive in real time. The advantages are obvious — the disk can be repaired and the computer isn't down in the meantime.

Additionally, data should be protected by producing sufficient back-ups, storing these back-ups in a clean, controlled, secure area and by maintaining full copies of all data, applications software and operating systems software off-site. Tapes

should be labeled and rotated at reasonable periods of frequency not to exceed one month in most cases. Tapes become brittle with age, therefore new tapes should replace older tapes every year. Operators should watch for problems and replace tapes or change brands if tape errors become more frequent.

4.2 Back-up and "Re" Recovery

Back-ups of data bases should be performed on a daily basis. Full system back-ups should be performed at least weekly. Multiple copies of each back-up should be done with one copy stored on-site, outside of the computer room, and a second copy stored off-site. If it is necessary to restore data from these back-ups, a copy should be made prior to installation. The primary reason for keeping multiple copies of data bases is to lessen the risk and to compensate for human error. If the majority of data is destroyed due to human error, the human error factor must also get built into the recovery process. Incorrect tapes are often mounted unintentionally in an effort to recover quickly, write rings are removed and valuable data may be destroyed, tapes or disks are many times rotated incorrectly thus destroying the initial back-up copy.

Maintaining multiple copies of history or year-end data is especially critical. If a disaster strikes, current data may be readily available for temporary recovery. However full recovery will most likely require historical data to be available for critical applications.

Test data should be backed-up and maintained off-site. This test data will be important during the re-recovery period to ensure all equipment and applications are operating as planned. There will be little time during recovery to develop detailed test data and expected results. As discussed in Section 7, this test data may be created by using a selected sample of live data.

Equipment and software must be kept in an area that is physically secure. Access to the computer room and the operator's terminals should be limited to operations staff. User personnel and analysts should not pick up listings or access the system via the computer room. Too many people, in a room full of millions of dollars worth of equipment, can accidently bump an important piece of hardware. Operator's terminals are normally configured differently from other terminals. If an unauthorized staff member attempts to access an application or run a job, he or she risks crashing the system or at minimum tying up the operator's terminal for some period of time. This policy will most likely not go over well with most analysts but is a definite requirement for physical security. The best way to ensure that unauthorized people are not allowed in the computer room is to provide some type of key-card access to operations staff. Any "non" operations staff requiring temporary access to the computer room, for such tasks as building or equipment maintenance, or to perform janitorial duties, should be monitored by operations staff. Most data center managers would be horrified if they discovered, too late, that the computer room floor was being scrubbed with soap and water or that two-way radios were utilized inside the computer room. Such negligence can result in damaged wiring or communication lines and scrambled

data on magnetic tapes.

In addition to keeping equipment secure, back-up tapes kept on-site or off-site should be kept in a secure location such as a fire proof vault. Combinations to such areas should only be known by a limited number of authorized staff members. Every attempt should be made to keep these areas temperature controlled. Although tapes and disks are not as susceptible as hardware to changes in temperature, extreme fluctuations in temperature can assist in the deterioration process.

A final consideration on the subject of physical security involves the recovery site. The Recovery Plan should thoroughly address the level of security necessary for the recovery period. This issue should be documented and tested so that if a disaster strikes, the physical security of the recovery site is not an issue. Access to the recovery site should be made available only to Readiness Team members. Keep in mind that Recovery Site personnel will have access to the recovery facility and that data security as discussed earlier becomes even more important.

4.4 Insurance

Insurance should be acquired for all equipment and hardware in the computer facility. Specialized insurance policies are available for data center facilities. A thorough inventory of all equipment software and data will be helpful in determining the extent to which insurance can be provided. Some companies have a global insurance policy which provides global coverage of equipment when acquired. Even so, a clear description of what is and is not covered should be documented within the context of the disaster recovery plan.

Requirements will differ depending upon whether equipment has been purchased or leased. As equipment gets older or outdated, standard insurance may not be available. A special policy may have to be tailored for such sites. Although you cannot insure against total loss of equipment, software and data should a disaster occur, the dollar amount of the potential loss should be documented in the Recovery Plan and the estimated loss should be discussed with top management and insurance company representatives. The investment in off-site storage, development of a plan and a recovery service should be considered the major insurance policy against any possible disaster. A fully tested Recovery Plan may, in fact, decrease insurance rates.

4.5 Microfilm & Hardcopy

For the most part magnetic tape or disk is not yet accepted as a valid media for official documents by the IRS or in court. Source documents are considered the only real proof that transactions either did or did not take place. Only within the last few years have IRS auditors begun to accept microfiche in place of hard copy source documents. In spite of the skepticism over whether or not microfiche or film would hold up in court, these media are useful and efficient in backing-up

hard copy documents. Microfiche or film may be stored off-site to protect critical data. Should disaster strike, this media is easily accessed and may be used to re-create lost data.

Hardcopy source documents or reports should not be overlooked as a source for recovering critical data. Although this virtually flies in the face of the paperless office concept, if tapes and disks have not been properly backed-up, hard copy may be the only available recovery source. In fact, users are often requested to re-enter data from source documents. Most data centers use this methodology to recover if data is lost or corrupted during regular daily processing prior to creating back-ups.

4.6 Vendor Committment

An important, yet often neglected, portion of the recovery planning process should address the relationship between the data center and hardware or software vendors. These vendors can be a great asset to your planning process and will become unquestionably required if a disaster occurs. The vendor should be requested to participate, at some level, in the planning process. If at all possible, a commitment should be received from the appropriate vendors for maximum consideration if a disaster should occur. A full recovery will be impossible unless software can be replaced or hardware re-ordered. Many vendors are incorporating disaster recovery services into ongoing maintenance agreements.

4.7 Personal Computers

Although most of this paper addresses the issues of planning, back-up and disaster recovery for large mainframe environments, the plan should also address personal computers. As MIS Directors, we spend thousands of dollars every year for PCs, software, printers and training. Some of our company's most valuable data resides in, or is massaged on, personal computers. If this equipment is destroyed or data is lost, a significant business loss results. For smaller businesses, the result is a disaster.

Many companies now realize that safeguards must be adopted to protect critical or sensitive data residing on PCs. A recent article in PC World[1] suggests that PC-based systems have an increased risk of being compromised. The article reports "with the growing number of PCs, everyone has access. Nearly everyday you hear of someone who has accidently reformatted a hard disk or deleted a database or directory." The article further suggests a number of preventative measures such as passwords, file locking, data encryption or use of sophisticated devices which prevent removal of floppy diskettes and, in some cases, set off an alarm. According to the article many experts recommend prioritizing data, attempting to place a dollar value on the data and then deciding how much money to spend on securing the data residing on PCs.

[1] PC World, March 31, 1987. "Corporations Wake Up to Data Security," Susan Janus, page 113.

Diskette back-ups should be made of all PC software. Back-ups should be made of all hard disks or data diskettes. These may be created using streamer tape equipment or diskettes. Copies should be maintained on-site in a secure environment with additional copies kept off-site.

Contingency site specifications should be included in the Recovery Plan and discussed thoroughly with the contingency site manager. If compatible PCs are not available at the contingency site, three options are available:

1. "Extra" PCs may be purchased and stored at the contingency site.

2. PCs may be leased for the recovery period.

3. PCs may be purchased at the time the disaster occurs.

Option 1 is the most preferable, although not always the most feasible. Option 3 is somewhat risky considering the time that may be required for delivery. Option 2 does not guarantee availability. One additional option, utilized by a Big-Eight firm, is to purchase several portable personal computers which are compatible with other company PCs. Qualified employees are encouraged to take the portables home for work-related or personal use. This offers the employee a much-valued "perk" and provides off-site storage for compatible PC hardware. If a PC disaster occured, the portables would be recalled.

The primary challenge of MIS Directors will be to provide planning for those companies which own a wide variety of non-compatible equipment. If possible, thought should be given to the compatibility issue as equipment is purchased. If a multitude of non-compatible equipment exists, more elaborate back-up capabilities must be developed. Each piece of equipment should be addressed individually to ensure full recovery possibility.

Section 5 — Contingency Site Description

5.1 Location and Specification of Contingency Site

As discussed briefly in Section 2, the location of the contingency site should be a prime consideration of the planning process. The ideal situation would allow each data center to maintain duplicates of equipment, documentation, software, communications and people onsite. The duplicate resources must then be backed-up off site for complete coverage. The cost of such an off-site center, normally called a "hot site," could be defrayed by selling services on the system to other sites. However, for most companies, this alternative is cost prohibitive. Resources are simply not available.

One alternative is to maintain a "cold-site" or an empty structure somewhere in another part of town. This cold site should be wired and prepped so that equipment can be moved in an installed quickly. The primary problem with a "cold-site" is that it only provides an empty building. Equipment and software cannot be tested. The primary advantage of using a cold site is that the phase-in period back to normal operations does not require a complete re-implementation. The cold site equipment can eventually be moved to a permanent location.

Another alternative is to purchase a contract with a service bureau located within reasonable proximity to your current location. This arrangement allows for convenient off-site storage and access to the facility for purposes of testing the Recovery Plan thoroughly. Although this arrangement is costly, it is not as expensive as maintaining your back-up equipment.

A fourth alternative is to purchase such a service from a service bureau located outside your immediate area. In this case, your Readiness Team may be required to travel to the service bureau location with current tapes or disks in tow. Some of these organizations provide equipment in roving vans which may be dispatched to a location of your choosing. Care should be taken in choosing this alternative. Customers aren't always afforded the opportunity to test plans at their convenience nor are they consulted on hardware or operating system upgrades. Systems must be reconfigured from the prior "disaster". Vans or trucks may break down enroute and not all service bureaus have fully tested recovery plans of their own.

The fifth, and least desirable, alternative is to develop a reciprocal agreement with other sites which have similar equipment and a similar configuration. This alternative, although better than no agreement, leaves a lot of room for problems. Even though the reciprocal site may have good intentions, the reality of such an arrangement is that most computer centers are already working at maximum capacity with any "spare" processing time available between midnight and 5 a.m. If processing time is available, your team may spend a great deal of time waiting in a que behind the low priority jobs of the reciprocal site. These sites rarely want to participate in the planning process for another site although it is only through this process that both parties can agree on terms. Configurations may change and equipment may be replaced or upgraded with little or no warning to the other party involved. If cost is the most important element, the reciprocal agreement may be the only alternative.

5.2 Nature and Terms of Agreement

Irrespective of what approach is selected (e.g. service bureau, reciprocal

agreement, etc.) a written agreement, jointly prepared and signed by both parties, should be developed. This agreement should cover the various levels of service to be provided in case of a disaster, the processing window available to the recovering site, facilities to be provided, such as office space or disk space, and other resources. A specific cost may be associated with each item or service level, or a fixed fee may be assigned which covers a specified level of service. If the agreement is a reciprocal agreement with no charge associated, both parties may want to include some type of clause to provide for reimbursement of expenses to the reciprocal or host site. Such expenses may include:

- Tape or disk charges
- Personnel charges such as overtime for host personnel
- Extra maintenance fees due to additional processing demands
- Supplies needed during recovery period

It is important that any agreement include specific descriptions and guarantees of availability of equipment .type, series and operating systems. Additionally, available processing windows must be specified and the system configuration must be documented for both parties. Estimates of the length of time needed for recovery conversion requirements should be developed and incorporated into both the agreement and the overall plan. This agreement should also specify a fall-back plan to be provided by the recovery site if for some reason the agreement cannot be fulfilled. The extra effort involved in developing a clear, concise agreement will be worth the pain if a disaster occurs. This agreement should be reviewed and signed by authorized personnel representing both parties.

A small company in Albuquerque has developed a creative approach to the reciprocal site. The company has, through a local users group, organized a group of compatible users who pay a fee. This fee covers the cost of a complete back-up system stored at the vendor site. The equipment will be made available to any participant should a disaster occur.

5.3 Configuration

In most cases, the hot-site you choose will have a "similar" configuration to your data center. "Similar," by definition, implies not exact. This means that some modification or reconfiguration of hardware, applications software, communications, or operating system may be required before reaching a minimum acceptable level of operations. This issue should be carefully considered in choosing a recovery site. If this reconfiguration is drastic and requires a reconfiguration daily to accomodate other processing requirements, another more compatible recovery site should be considered.

In any case, a copy of all of the configuration specifications, for both the disaster site and the recovery site systems, should be included in detail within the recovery plan. All ongoing changes to the configuration requirements for your site or the hot site must be incorporated into the Recovery Plan as these changes occur. A mechanism should be developed to coordinate this. A numbering convention, similar to those often used for updating documentation, should be used to ensure the most current version has been received and incorporated.

In addition to including copies of all configuration specifications, procedures should be developed which provide specific tasks to be carried out for

reconfiguration of the hot-site system to accomodate both the subscriber and recovery site. These procedures should take the form of a detailed cross-walk from the hot-site configuration to the required recovery configuration and from the disaster site procedures to the recovery site procedures. These procedures should be fully tested at the hot-site by the Recovery Team. An important aspect of reconfiguration that should not be overlooked is that a service bureau or traveling site will most likely be configured to accomodate the last site at which a disaster occured. The agreement or contract with such a hot-site should specify a "base-line" configuration to be provided by the hot-site.

5.4 Accomodations for Staff/Control Center Operations

As discussed in Section 2 of this paper the contingency site or hot-site chosen should provide work space for various members of the Recovery Team. The amount of space should be directly related to the number of people assigned to the Readiness Team and the processing windows available from the contingency site. Access to programmer, user, and operater designated terminals will be required. These accomodations should be reviewed thoroughly before any agreements or contracts are signed.

If the agreement is a reciprocal agreement, there may be some resentment or feeling of intrusion from the reciprocal site staff when the Readiness Team arrives. The contingency site manager should discuss all recovery accomodations at the time the plan is developed so that all staff members will be fully aware of such potential problems and any resentment may be defused ahead of time. Remember, the Readiness Team will be working under stressful circumstances in "Panic" mode. This atmosphere will carry over to the contingency site staff, especially when work space or resources must be shared. If at all possible, Readiness Teams should meet with contingency site managers, analysts and operations staff frequently to ensure some level of working relationship prior to the occurance of a disaster. An effective agenda might take take the form of a bi-monthly meeting updating both parties on configuration changes, anticipated hardware or software upgrades, and changes in procedures.

5.5 Scheduling

Processing time, print time and use of facilities must be carefully scheduled to ensure a recovery that is as smooth as possible. This scheduling process will, no doubt, be significantly different than the process currently utilized for standard operations at your site. If you have choosen a service bureau or reciprocal contingency site, other processing schedules must be factored into computer usage or scheduling needs. A contingency site, no doubt, has scheduled downtime for such things as maintenance. This downtime must be factored in to your recovery scheduling parameters.

A good starting point for developing recovery parameters is current processing schedules (daily, weekly, monthly and yearly). These schedules, combined with the list of prioritized applications, should be discussed with the contingency site manager in an attempt to narrow the available processing window and scheduling parameters. A separate recovery "base-line" schedule should be developed and incorporated into the recovery plan with the knowledege that this schedule may need to be refined or modified at the time of recovery. Since a disaster is an unplanned event, a final schedule cannot be developed. It should

be noted that the intention in developing a "base-line" schedule is to anticipate the possible scheduling problems. The goal in this process should be to standardize, make reasonable assumptions and to plan for as many situations as possible.

5.6 Procedures and Documentation

A major assumption of the disaster recovery planning process is to ensure that the data center has a fully tested set of concise procedures and full set of user, program and operations documentation which is updated on a regular basis as changes are made. If this assumption is not true, time should be allocated during the planning process to develop a minimum level of documentation. This documentation should include:

- User guides for each application
- Program documentation for each application
- Data base schemas
- Data dictionaries
- Operations guide
- Sign on and security documentation
- Hardware configuration parameters
- Applications-software configuration parameters
- Operating system software configuration
- JCL, by application and program
- Job streams, by function and program
- Name, address, phone numbers of vendors
- Name, address, phone numbers of clients
- Name, address, phone numbers of users

An extra copy of each of these items should be kept on-site separate from your data center or computer room. The second copy should be maintained off site with back-up tapes and disks. If you are purchasing Disaster Recovery services from a service bureau, you may be able to store documentation and procedures manuals at the contingency site. A current copy of the Recovery Plan should also be stored with this set of documentation. A disaster would be complicated further if a copy of the plan were not available even though all of the thought-process had been performed and every possible event had been anticipated and documented.

Security procedures should be given special attention during recovery planning. If key passwords are lost or destroyed and not known to the Readiness Team, applications could be fully recovered but inaccessible from the contingency site. These passwords must be kept in a secure place offsite so that non-authorized personnel do not have access to these codes or capabilities. The best protection would be to store these passwords on a tape which may be indepdently accessed by two of the key members of the Readiness Team.

Another aspect of security which must be considered is data security during the recovery period at the contingency site. Contingency personnel must protect other data residing on their system from unauthorized access. Your data base manager or security specialist will have some concerns in letting contingency site personnel have authority over where data is located and how it is accessed. Operations staff of both data centers will fell protective of their procedures and/or equipment. Although many of these problems can be avoided by providing

program and operations documentation which is updated on a regular basis as changes are made. If this assumption is not true, time should be allocated during the planning process to develop a minimum level of documentation. This documentation should include:

- User guides for each application
- Program documentation for each application
- Data base schemas
- Data dictionaries
- Operations guide
- Sign on and security documentation
- Hardware configuration parameters
- Applications-software configuration parameters
- Operating system software configuration
- JCL, by application and program
- Job streams, by function and program
- Name, address, phone numbers of vendors
- Name, address, phone numbers of clients
- Name, address, phone numbers of users

An extra copy of each of these items should be kept on-site separate from your data center or computer room. The second copy should be maintained off site with back-up tapes and disks. If you are purchasing Disaster Recovery services from a service bureau, you may be able to store documentation and procedures manuals at the contingency site. A current copy of the Recovery Plan should also be stored with this set of documentation. A disaster would be complicated further if a copy of the plan were not available even though all of the thought-process had been performed and every possible event had been anticipated and documented.

Security procedures should be given special attention during recovery planning. If key passwords are lost or destroyed and not known to the Readiness Team, applications could be fully recovered but inaccessible from the contingency site. These passwords must be kept in a secure place offsite so that non-authorized personnel do not have access to these codes or capabilities. The best protection would be to store these passwords on a tape which may be indepdently accessed by two of the key members of the Readiness Team.

Another aspect of security which must be considered is data security during the recovery period at the contingency site. Contingency personnel must protect other data residing on their system from unauthorized access. Your data base manager or security specialist will have some concerns in letting contingency site personnel have authority over where data is located and how it is accessed. Operations staff of both data centers will fell protective of their procedures and/or equipment. Although many of these problems can be avoided by providing clearly defined recovery team member job descriptions, security procedures should be developed specifically for the recovery period. These procedures should be developed in conjunction with contingency site staff and incorporated into the Recovery Plan.

clearly defined recovery team member job descriptions, security procedures should be developed specifically for the recovery period. These procedures should be developed in conjunction with contingency site staff and incorporated into the Recovery Plan.

Section 6 Recovery Procedures

6.1 Team Assignments

Section 1 addresses the nature and make-up of the Readiness Team. The focus of defining a Readiness Team was on minimizing risk by developing job descriptions and detailed project schedules for the recovery period. The team will most likely be made up of people from your organization. However, as mentioned previously, there are cases where outside consultants should be considered.

The team will be responsible for responding to a critical situation in the earliest stages following a disaster. The team leaders will be required to create organization out of chaos. The most effective way to ensure that some organization or structure will be imposed on an otherwise chaotic situation is to develop detailed task plans in a similar format to those required for development projects.

These task plans should include an introductory statement describing the purpose and goals of the major task groupings. These statements should be clear and objective. Esoteric statements about "doing good things for users while upholding data processing standards" have no place in the task plans. Major tasks should be defined in terms of functional groupings such as "restore database". These functional groupings or tasks should be further reduced to subtasks and steps. Key recovery personnel should be assigned responsibility for each task. Estimated hours should be assigned at least at the sub-task level.

Valid "manhour" estimates may be developed based on at least four sources:

- The length of time required for testing each particular task and subtask
- The amount of time the planning team has determined your company can survive without access to critical data
- The amount of time the planning team has determined will be required to gain access to a new facility or new equipment
- The time required to reconfigure and test any new aquisition

Although these estimates may not be completely valid, we in the MIS field are experienced in generating reasonable estimates for new development projects. These estimates are based on contingency planning for unknown problems. Contingency planning methodology should play an important role in developing detailed task plans for Disaster Recovery purposes.

6.2 Implementation of Procedures

As discussed in Sections 3 and 5, detailed procedures are a necessary part of the recovery plan. If, however, these procedures are not implemented correctly the plan will be of little use.

Readiness Team members should participate in developing the plan and writing the recovery procedures if at all possible. Team members should also participate in updating procedures, when changed, to ensure maximum familiarity with procedures. New staff or alternate team members assigned to the team should be required to review the procedures as a regular part of an ongoing training program.

It should be re-emphasized that although standard procedures are used as a base-

-line for development of recovery procedures, disaster recovery is a non-predictable and non-standard situation. Therefore, additional procedures must be developed to handle emergency situations such as:

- Contacting and informing Readiness Team members about the disaster
- Notifying contingency site and off-site storage facility
- Notifying Top-Management
- Contacting and preparing user departments
- Informing MIS staff who are not assigned to the Readiness Team

These procedures, along with the technical procedures for activating the plan, are primarily focused on attaining an acceptable level of operation of critical systems. Separate procedures should also be developed for the replacement of the data center and eventual return to normal operations. These procedures should be developed in as much detail as possible and task assignments should be made to non-readiness team members. Although secondary to emergency recovery, replacement procedures should be given an appropriate amount of attention. The best format to use in the development of a replacement plan would be an acquisition plan or implementation plan utilized when the original hardware or software was installed.

Although the exact configuration may not be available, the procedures should include how and where to contact vendors, the location of the vendor's plant from where the hardware or software will be shipped and alternative compatible versions which could be utilized if current products are not available. These procedures should be reviewed with the appropriate vendor representative to ensure maximum cooperation. Most vendors will agree to assist clients during the recovery period for obvious reasons -- these situations ensure an immediate sale. The specific support offered by a vendor can be incorporated into your ongoing maintenance agreement. Even if there is some level of dissatisfaction with your current systems, it is not recommended that the recovery period be used to acquire new and different hardware or software unless the disaster can be directly linked to the hardware or software in question.

6.3 Activation of Control Center

A control center is usually the place from where recovery operations are managed and information is filtered and distributed for the recovery period. The primary purpose of a control center is to provide a central location from which the whole recovery process can be managed. An important aspect of setting up a control center is location. If the disaster is due to "natural" causes such as flood, fire, earthquake, etc., it will be impossible to locate the control center at the site which has experienced the disaster. Although the control center may be located at the contingency, or hot-site, consideration should be given to locating this center in close proximity to the disaster site without physically locating at the hot-site. This will allow a primary work location for the replacement team members and an alternative work location for Readiness Team members. Such a separation of the location of the control center from the contingency site will be essential if a reciprocal site is being utilized for recovery and sufficient space is not available for the Readiness Team on a full-time basis.

6.4 Feedback Loop with Management

It is extremely important to develop a clearly defined mechanism to provide

top management with progress and status reports and to receive instructions or progress reports from top management. It is critical that the information disseminated be precise and controlled. If the disaster is severe enough, top management may need to prepare a press release. Incorrect information could create a second disaster or at minimum result in awkward press coverage. In turn, the readiness team needs to be informed on the progress of a replacement facility or replacement equipment.

It is equally important to define a mechanism to provide progress reports to user departments. As discussed in Section 1, the best way to accomplish this feedback is to assign knowledgeable users to the Readiness Team. A major responsibility of these team members will be to coordinate with, and inform, user departments on the ongoing progress and efforts involved in recovery. The user departments will be anxious to learn of any progress. Hardware and software specialists should not be subjected to the additional stress involved with defending recovery center positions. User liaison's are better qualified to explain the conditions and target dates for full recovery.

6.5 Activation of Contingency Site

The entire process of activating the contingency site should be repeatedly tested to ensure all parties involved are trained and knowledgeable about emergency procedures. The more frequently these "fire drills" take place, the more routine the process will become. The following process should take place:

- Readiness team members should be contacted
- Contingency site should be contacted
- Back-up tapes, disks, and documentation should be picked up by appropriate team members from off-site storage locations
- Processing schedules should be verified
- Status report should be given to top management
- System should be reconfigured
- Critical applications should be installed
- A back-up should be taken prior to any processing
- Critical data should be loaded
- Partial or trial runs should be accomplished to verify system processing
- Recovery procedures should be initiated based on system priorities

If the statistics available are realistic, all of this process must take place within two to 48 hours, depending on how critical the system is to the company's survival.

As is clearly apparent, the proximity of the contingency site is an important factor when considering activating the contingency site. If back-up tapes are located on one side of the city and the contingency site is located on the other side, the time required for this task increases substantially. If the chosen contingency site is located in another city, the time to complete all of these start-up tasks is additionally increased.

An equally important factor in activating the contingency site will be whether or not changes to the Recovery Plan have been tested. Additional time may be required if procedures are changed dramatically during the activation period.

6.6 Implementation of Replacement Data Center

Although this paper focuses primarily on the recovery period and in sustaining an acceptable processing level, an equally important major task area includes the eventual replacement of the data center facility as well as any hardware or software. It is therefore important that full documentation is available describing the current facility including:

- Architect's plans:
  - -heating and cooling
  - -electric and power
  - -sprinkler and halon
  - -elevations
- Equipment positioning
- Complete inventory of:
  - -hardware
  - -software
  - -communication devices
  - -network

This documentation should include dates of purchase, serial numbers, series numbers, version numbers, size and quanities of all hardware and software. Some logical baseline documents to use in preparing this documentation include:

- Original invoices or lease agreements
- Marketing literature
- Packing slips

A drawing should be created which shows details of the site including specific locations of equipment. A cross reference to serial numbers and other identifying information should be created to fully document these resources. Specifications for the software configuration should also be created and included in the plan.

If full replacement is required, this documentation will also be used to design a new data center facility and to order replacement equipment or software. It should be noted that this replacement design and ordering process should begin simultaneously with the efforts to sustain an acceptable processing level. The contingency site must be used until a replacement data center is available and replacement equipment is delivered. Time must be allowed for site preparation to ensure all aspects of operation are feasible.

After the replacement center is fully prepared, a reasonable period should be allowed for parallel processing. Less trauma would result from extending the parallel period than from re-implementing the recovery plan at the contingency site too quickly. The same considerations should be given to this parallel period as would be given to parallel processing as a result of converting from an old system to a new system. The parallel period should only involve critical applications which were required for the recovery period.

6.7 Phase-in of Return to Normal Operations

After the replacement data center is prepared and systems are running parallel, plans must be implemented to phase-in non-critical functions and to prepare for return to normal operations. The list of critical applications prepared during

the planning process, and discussed in Section 2, should be utilized. Applications and processes should be phased-in according to their ranking of priority. Care should be taken in this process due to the fact that some of these applications will have not been utilized for a significant period of time. It is highly recommended that non-critical applications be installed in a test environment utilizing test data until it is determined that the application is functioning properly. Many of the same procedures discussed in Section 6.5 used to activate the recovery site can be used to return to normal operations. If a hardware or software change was necessary to implement the replacement site, the appropriate modifications must be made to procedures to ensure all systems have been considered. It is highly recommended that these non-critical applications be installed in a test environment utilizing test data until it is determined that the application is functioning properly.

Section 7 - Testing and Ongoing Maintenance

7.1 Test Plan

A recovery test plan should be prepared for all applications. The primary purpose of such a plan is to develop detailed procedures for periodic testing of the contingency site to ensure smooth conversion in case of an emergency. The secondary purpose of such a plan would be to provide test procedures and test data for the phase-in period prior to returning to normal operations.

The test plan should include detailed procedures for testing the functional area of each application or procedure utilized. In addition, the test plan should provide test data and expected results by function to be tested by application. This test data may be compiled from test data prepared for the original implementation of each system or may be created from live data currently residing in the system. Source documents may be collected for each major transaction or functional grouping. Reports or screens may be printed for the valid or expected results.

As with other portions of the Recovery Plan, the more detail that is supplied in the test-plan, the easier it will be to implement if a disaster occurs. Like other plans, the test plan must be maintained and updated on an ongoing basis as new systems are acquired or current applications are modified. It makes sense to assign the test plan maintenance tasks to the analysts responsible for each application. It is therefore also important to note that these analysts may not be assigned to the Readiness Team. The plan should therefore be clear and concise and Readiness Team members should receive up-dated copies of any changes as the changes are incorporated.

7.2 Policies and Procedures

The majority of well-run companies or MIS organizations maintain some level of documented policies and procedures. These documents, unfortunately, are not always updated or revised on a timely basis. Attention should be given to updating these documents in consideration of disaster planning. Additional procedures and policies will most likely be identified and added during the recovery planning process. The maintenance and update of these documents is an extremely important part of recovery planning. If procedures are incorrect or not updated, more confusion will be added to an already stressful situation if a disaster occurs.

Although organizing and updating policy and procedure documents is not the most exciting task in the MIS field, it is certainly one of the most critical tasks given the overall goals of disaster prevention and disaster recovery.

7.3 Review and Update

The greatest portion of a Disaster Recovery Plan must be based on the short and long-term goals of the company. As the company grows and changes, the Recovery Plan must be updated to reflect these changes. In order to ensure full understanding of company goals and plans as they affect the Recovery Plan, a review session should be held periodically with top management. This review may be incorporated into regularly scheduled steering committee meetings but should be treated as a separate and distinct agenda item. The priority of critical applications may change as the company grows or changes, new systems may

redefine priorities.

The readiness team should not wait until these meetings with top management to update the Recovery Plan. Although the ideal frequency for incorporating updates is whenever a change occurs, a more feasible frequency is monthly. A deadline can be imposed by which time all updates must be incorporated into the disaster plan. The Readiness Team leader, the MIS director and the Operations Manager should review these changes to ensure all updates are properly incorporated.

Conclusion

In conclusion the Disaster Recovery Plan is an ever changing document. If developed properly, the plan may actually assist in preventing disasters and will certainly assist in standardizing and organizing critical business areas. Most of the disasters that occur can be prevented. The majority of disasters are not caused by flood or fire, although these are the most highly publicized issues. Hardware or software defects and human error are the leading initiators of situations which may be defined as disasters. Disasters are not caused by hardware or software failure alone. Disasters are caused by not being able to recover critical data and functions within an acceptable period of time.

Although well organized shops with a full set of current documentation will be in the best position to prepare and refine, and ultimately implement a disaster recovery plan, any progress towards the completion of the plan will be helpful at recovery time.

"@", "*" 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.

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 (DBCB) 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 occurence 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 *full record* 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.

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 = Offset(F+1) - 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.

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.

DBPUTs 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 user is verified to have write access to the field and 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.

USING THE HP3000 TO PREDICT THE FUTURE

Otis A. Whitehurst
Vermont Housing Finance Agency
One Burlington Square
Burlington, Vermont 05401

ABSTRACT

This paper explores the use of a forecasting model on the HP3000 and describes a case history of a simple cash flow analysis we developed to evaluate the value of loans to be packaged for creating a secondary mortgage market. Included are a subroutine for calculating loan parameters when any of Loan Amount, Interest Rate, Term or Payment Amount are known.

FORECASTING

Forecasting is the art of assimilating known information into some form of model that can be used to estimate future events. As long as the assumptions made in the model are accurate and stay in effect, the model will "predict" the future for a limited scope. Forecasting is a common experience in our every day lives. The principles are inherent in many activities such as predicting the weather, budgeting the spending of your paycheck, controlling the traffic signals in a community, buying groceries for the week, and deciding on whether to put gas in your car. Often we evaluate a situation intuitively and forecast as a matter of course without needing to consciously construct a model and do calculations.

Take the example of deciding on whether to put gas in your car. Depending on your financial status and personal preference several components come into play. A whole series of questions are informally posed and answered. How much gas does your fuelgage indicate ? When will you next have the opportunity to get gas ? How far will you need to drive before that next opportunity ? How many miles per gallon does your car get ? Your car gets more miles per gallon in highway driving than in city driving. What proportion of the miles you expect to drive will be on the highway and what proportion will be in the city ? How many gallons are in your tank based on the amount indicated by the gas gage ? Do you have time to get gas now ? Do you have the finances to get gas (cash or credit card) ? Should you fill the tank or get a set amount of gas ?

Seems rather complex doesn't it. Yet most people make the decision in a moment. You can imagine how the situation would change if a study was made and revealed that much money and time could be saved if certain principles were followed. Someone might even devise a short course or seminar on how to most effectively decide on gassing up the automobile. An onboard computer could accept the parameters, calculate the projected driving load based on past experience and give a recommendation as to whether to fill up. I can see it now. Sitting in the emergency lane on the expressway while the other cars whiz by. "Out of gas ? But the computer said I didn't need to get gas yet !".

Let's extend the problem a little further. Suppose instead of the family car you are responsible for a fleet of cars or trucks or airplanes. You must set up a schedule for refueling and maintenance that will keep the vehicles in service and keep costs in line. Instead of each driver deciding when to get gas or schedule maintenance, maybe you should use forecasting and issue guidelines.

BUSINESS USES

As you can see by the examples given above, forecasting is an integral part of decision making. Good forecasting (some say lucky) will result in information that can help in achieving the goals you establish. Poor forecasting will result in inaccurate or misleading information that can be a hindrance to achieving your goals rather than an aid. Businesses have used forecasting in their decision making as long as there have been businesses. The very decision of starting a business involves forecasting. The process may be formal or informal, but in any case the risk is compared to the potential gain. If the potential gain is worth the risk, a business is launched.

Forecasting is commonly used in business for inventory management, sales analysis, business plans, project management and cash requirements. If the "business" is an academic one, forecasting can be used in scheduling courses and instructors, allocating resources, managing the physical plant or assigning dormitory space. As you look at your own working situation, you can probably think of many more applications that use forecasting.

As in fashion, forecasting experiences periods of time where some methods are more popular than others. Generally, the particular fad will be put forth with a lot of promotion and fanfare. Many vaild reasons underlying the forecasting method are detailed and the benefits are expounded. Take for example economics. Over the past few years "Supply Side" economics has been in fashion. Forecasts were prepared and whole policies were based on the forecasts derived from the principles that underlie "Supply Side" economics. In the public sector, there is the additional complication of politics.

Have you ever heard of the 80-20 principle ? I first read of it as applied to inventory management, but have since seen it applied in several other areas. The basic premise goes something like this: "Twenty percent of the items in inventory are responsible for the bulk of your sales, while eighty percent are responsible for the balance". The implication is that the twenty percent should be managed more agressively than the other eighty percent. Other variations on the theme include: "Twenty percent of the items are responsible for the bulk of the money tied up in inventory", "A certain limited number of items in inventory are responsible for 80% of your sales, while the other items are responsible for 20 % of sales". Depending on the particular version you subscribe to, your forecasting methods for inventory requirements will be structured differently.

The method of forecasting used in a business will ultimately affect its profitability. This fact often becomes obscured due to a variety of reasons. First, the results of decisions based on a forecast may not be measured well. This can be due to poor planning, problems in quantifying the results, or poor implementation of the monitoring system. The results of decisions based on forecasting may be obscured by time and the ensuing changes in the business environment. You may not be able to determine if the forecasting method was faulty or if unforseen complications altered the outcome. There may not be a structure in place for periodically evaluating the forecast and modifying your model to more accurately reflect reality. The computing environment you use or the way that you use it may not allow you to modify your model in a timely manner. An added complication is time and money pressure. Few companies have the resources to allow detailed analysis to formulate forecasts, much less to determine if they work and how they might work better.

For example, in the case of managing a fleet of vehicles, you would need to institute a system to monitor the fuel usage and maintenance expenses to evaluate your effectiveness. If the monitoring system is flawed due to negligent reporting or other factors, you may not be able to determine a clear improvement or worsening. In addition to your monitoring system, the time factor could alter the picture. Your company may add or remove a large number of vehicles from service so that the original assumptions are no longer valid. The quality and cost of fuel or lubricants may have changed drastically. Think of the problems faced in fleet management over the last twenty years. In the areas of fuel usage and maintenance there have been few constants. Vehicles have changed. Fuel prices have jumped and fallen back. Fuel rationing was in effect for a time. The maximum speed limit was reduced across the whole country and is now increasing again in many places.

FUTILITY OF FORECASTING

What's a poor forecaster to do ? The best you can with what you know at the time! NO ONE (that I know) CAN PREDICT THE FUTURE RELIABLY. Your forecasts are doomed to failure before you start. Some are more doomed than others. The best you can hope for is to structure your forecasts to accurately model the real life situations that are important to your forecast. If you can clearly identify the goal of your forecasts and the factors affecting it, you have a greater chance of success. In addition, if you can structure a system for timely feedback and changes in your forecast, you may lessen the impact of the items you did not take into account and of changes in your business environment.

THE GOAL

What do you want to achieve anyway ? Take a basic business plan. Is your purpose to secure financing and investors or to layout a working outline for the business ? The two differing goals result in a different emphasis. You may have to answer to investors when they confront you with the discrepancies between your prospectus and your annual report or you may have to live with the pitfalls that were not taken into account in your working

outline. Is the goal of inventory management to keep as small an inventory as necessary ? What about customer service and the cost of backorders ? Goal setting is necessary before you start. You may find that the most obvious or initial goals are in conflict with other underlying goals of your business. Take the time to sit down and think through the purpose of and expectations for your forecast. A little time spent in planning will save a lot of time in implementing and refining later. Goal clarification is especially beneficial in helping the individuals involved in the design and development of the forecast maintain focus as the project progresses.

THE TOOLS

I like the HP3000 as a tool for forecasting. It is flexible and powerful. The HP3000 can aid in the collection, storage, retrieval and evaluation of the information you must manage to produce effective forecasts. If used properly, it can insure that the overhead of managing that data is minimized and allow timely changes to your forecasts. The IMAGE database environment assures the integrity of your information and allows you to access it in a variety of ways. You can easily download information to a personal computer for analysis in spreadsheets or you can use the HP3000 for the analysis.

TECHNIQUES OF FORECASTING

This paper is intended more as a study in a particular case of using the HP3000 in forecasting rather than as an exhaustive description of all forecasting techniques. The selection of a particular technique is heavily dependent on the information available and on the underlying real world situations you may be interested in predicting. In broad terms, forcasting may be broken into at least two groups. The first might be called statistical forecasting, while the second group might be called analogy or model based forecasting.

Statistical forecasting includes probabilistic inventory modelling and all forms of random or partial group sampling where the information is extrapolated to project probable future events or situations. The underlying assumption is that samples of a portion of the population you are interested in represent the total population within an error margin which you can calculate.

Analogy or model based forecasting attempts to constuct a model on the computer that functions similarly to the real life situation. A simple example of this is a program that creates an amortization schedule for a fixed rate loan. The amortization schedule models the payments made and the amount of principal and interest over the life of the loan (provided that the payments are made on the due dates).

Most models will need to use a combination of these two methods to achieve satisfactory results. In the example of fleet management, the statistical method might be used to determine one or a few "typical" vehicle usage patterns and estimate the gasoline consumption and maintenance required for vehicles classified under the predetermined usage "classes". The modelling method

would be used to project a target date for maintenance of a specific vehicle by figuring the mileage the vehicle is projected to be driven for the work schedule planned while taking into consideration the workload of the maintenance shop, the time required to perform the maintenance, and the availability of personnel or other resources to perform the maintenance. You know the route this vehicle must travel is 75 miles long and the vehicle is driven five days per week. Your maintenance shop can service 10 vehicles a day. Given the facts, you can determine a suitable time for maintenance.

Good forecasting falls somewhere between the seat of the pants approach and the accepting what the computer says as law approach. Just remember, intuition can be wrong as can the figures a computer spits out. Real life situations defy absolute prediction by mathematical models.

Reliable forecasting software should draw on as much real life information as is feasibly possible. The assumptions that are inherent in the forecast should be documented and the people relying on the forecast should be made aware of the underlying assumptions. A method of feedback should be incorporated into the forecast for future correction in the event the underlying assumptions are not correct. When it is likely that some of the underlying assumptions may change significantly between the time of the forecast and the actual future event or situation, it is wise to include a range of potential values in addition to the predicted value. If it is appropriate, a table of potential outcomes is more useful (and safe) than predicting "A" specific occurrence.

UNREALISTIC EXPECTATIONS

It is unfortunate and unrealistic, but many companies want the computer to tell them what to do. "I just want a number, not a page full of figures and contigencies." They have a fantasy forecast in mind that bears little resemblance to what can be delivered. This may not become apparent until well into the development process. Avoid unrealistic expectations whenever possible by involving the ultimate consumers of your forecast in the development process.

A classic example of unrealistic expectations is in software development. We all have had the question posed (possibly quite frequently): "When will the program be ready?" or more likely, "When will the new software system be online?". Have you ever promised completion by a certain date and failed to meet your deadline? Have you ever met a deadline that you promised? Program development is a tricky forecasting problem. Involving end-users in the development process by using prototyping is helpful. The users see progress and gain a little better understanding of the work involved in producing good software. I do not have any pat answers or tricks to use for better forecasting in program development. Figure your best estimate and double or triple it. If they get a pained expression when you tell them how long it will take, cut your estimate back by a qualified 25% (maybe if all things go well).

A CASE HISTORY

Recently we have been evaluating the possibilities of purchasing Community Development Block Grants from towns in Vermont. The towns had previously received CDBG money and made loans for purposes that were consistent with VHFA's guidelines. The loans were made at low and moderate interest rates to create or rehabilitate housing in the communities. Some of the towns felt that it would be more productive if they could find a way to get the money back faster than waiting 10 or 20 years, and reloan the money for further housing uses. VHFA's role would come in acting as an intermediary to package the loans from enough towns to create a pool of loans large enough to be feasible as collateral for some form of security such as a bond or note. A note or notes would be sold with the proceeds to be used to purchase the outstanding eligible CDBG loans from the towns at some discount from face value. The income from the payments on the loans would be used to pay interest on the note(s) and eventually to pay the note(s) off.

At Vermont Housing Finance Agency (VHFA), most of our activity is oriented toward selling some type of security such as bonds or notes and using the proceeds to purchase mortgage loans in the state of Vermont. We normally operate within two major constraints. First, we must be able to structure a deal that works financially. The income from mortgage payments must be able to cover the interest payments to our bondholders plus our operating expenses. Our housing programs do not normally receive any state funding or subsidy, so the programs must be self-supporting. Usually the bonds that we sell are tax exempt. The second constraint is unique to sellers of tax exempt bonds. We are allowed to make only a certain amount above the rate we pay on our bonds. No buying low and selling high for housing agencies. We have to structure deals where the arbitrage (difference in rates between what we pay and what we get) is capped. Our rate of return is limited by statute as well as what the market will bear. Any deal that is structured must fall in the range between the tax exempt bond market, prevailing mortgage rates, prevailing investment rates for our reserves and the arbitrage limitation.

Though the constraints for Housing Agencies may be an obstacle to forecasting, there are benefits for the bean-counting portion of forecasting. Since we deal mainly in dollars rather than hardware inventory, systems are in place to measure the impact of decisions based on forecasting. Our inventory is uniform (dollars and cents). We use a form of accounting called fund accounting where each bond series that is sold is accounted for separately from the others. The mortgages that we purchase are identified with specific funds. We have funds for debt service, funds for paying the bonds off when they become due, revenue funds and others as needed. Our accounting department knows where every dollar is and how it got there. They also are acutely aware of which assumptions do not pan out.

Our forecast of the cash flow involved in purchasing and servicing a pool of mortgages with varying maturities, terms and interest rates uses a combination of the statitical and modelling

methods. We characterized the nature of payments on mortgages and for bonds based on our past experience and projected future events (transactions) for a known group of mortgages.

If we had the luxury of travelling forward in time then back again, the process would be simplified, but still not simple. Even if we could accurately predict interest rates, demand for our mortgage money, default rates, etc., we would still need to structure a system where the money coming in was able to cover the money needed to pay debts. The amount received varies from month to month as does the amount disbursed. There is no formula for calculating the proper structure. Statistical analysis isn't worth a lot when Mr. Volker decides the federal reserve rate needs to be lower. What do we do? We make an educated guess. Horrors! An educated guess? Well an experienced educated guess blessed by bond counsel and other official persons, but yes an educated guess. Oh, by the way, provisions are generally included to limit the impact of wrong guesses such as reserves for losses, mortgage insurance, and callable bonds. Housing Agencies and many other financial entities are usually very conservative in assuming lower than expected rates of return on reserves and higher than expected rates of delinquent loans when producing forecasts and cash flows. Hopefully the safety net is not needed, but it is available as a precaution. In the fleet management problem you will not be hurt terribly if a vehicle has to go in for maintenance a day or two later than projected due to a miscalculation, but in finance, creditors tend to get bent all out of shape when you are a few days late on a couple of hundred thousand due in interest because of a cash shortage.

Let's consider the trip to the future. If you were allowed to go into the future you could look over VHFA's books and detail every transaction that affected the fund accounts for this bond series. You would be able to see how the funds from the bond proceeds were disbursed for mortgages and expenses. You would be able to see how the reserves were invested and how much income was collected from short term (non-mortgage) investments. The activity of the loan portfolio would be obvious as mortgage payments were made and as mortgages were paid off. When you returned to the present, you could give enlighted advice on what to expect.

TRANSACTION BASED MODEL

We can predict with a reasonable degree of accuracy the effect of a transaction. We can generate possible scenarios of transactions on a hypothetical set of funds, add up the cumulative effects of a lot of transactions and evaluate the results. We can set up formulas for what will occur with a given set of conditions. The guessing comes into play in deciding what the prevailing conditions will be.

Let's look at some of the transactions. We will not detail every single transaction, but will examine enough transactions to give an understanding of how to perform transaction based forecasting. The computer will not give us the "right" answer, but will act as a tool or filter to help us evaluate the alternatives.

Step one: sell some bonds. Simple, right? How much should we sell? $25 million? $100 million? Who is going to sell them? We have to evaluate several factors including the amount we are allowed to sell, the prevailing rate for organizations with our bond rating, and the demand for qualified mortgages of the type allowed under these bonds. We have good bond counsel and a good underwriter. Their fees must also be considered in the structure of a program. Fees for floating a bond issue are not negligible. What types of bonds should we sell? What maturities and interest rates should they bear? The possible combinations are almost endless.

Step two: Buy some mortgage loans. How many can we buy with the proceeds from the bonds? What interest rate should we offer? We are only allowed to charge a set maximum above the rate we pay on the bond we sell. Should we charge less than the maximum? What rates are available for mortgages in the conventional (non-tax-exempt) market?

Step three: Collect payments. This one is easy. Everyone pays the correct amount every month. No late payments, no loan defaults, no pre-payments, no one sells their house before the mortgage is paid off (twenty-five years).

Step four: Pay our bondholders. This one is a little more predictable. They expect payment on the day due in the amount agreed to previously. Unless of course we sell callable bonds in which case we may want to consider buying some bonds back in addition to paying the interest due if our cash reserves are higher than expected.

As you can see from the above steps, there are a lot of factors to consider in this forecast. There are companies whose entire business is based on making forecasts for bond cash flows. At VHFA we still use an outside company to do the projections for our bond issues. This less sophisticated projection in house was used as a planning aid to narrow down the field to realistic potential rates and amounts.

The goal in the purchase of CDBG loans was to structure a deal that would give the maximum amount back to the towns while covering the expense of the security sold and administrative expenses. The towns submitted a list of their outstanding loans so that we had a known pool of loans that we could potentially purchase. We had an idea of what to expect in the area of delinquencies and payoffs.

Our first step was to layout a general structure that was acceptable to both sides. We assumed that all potential loans would be purchased. Since we had the vital information on each loan at a known point in time, we were able to estimate the loan balances for future potential purchase dates. There was a small variation in the raw information reported from the towns. Some reported the original loan amount, term and interest rate, while others reported the original term and rate with the current balance of the loan as of a certain payment date.

We used the following subroutine to calculate the needed information from the data given. As long as any three of the four components (Payment Amount, Interest Rate, Term, Loan Balance) were given, we could calculate the missing component. The term for these loans was measured in months and the interest rates were fixed for the life of the mortgages.

```
C*****************************************************************C
C*****************************************************************C
C                                                                 C
C       LOANCALC - SUBROUTINE TO CALCULATE LOAN AMORTIZATION      C
C                  PARAMETERS.  ANY THREE OF THE FOLLOWING        C
C                  FOUR COMPONENTS MUST BE SUPPLIED:              C
C                  1. PAYMENT - AMOUNT OF P & I PAYMENT           C
C                  2. LOANBAL - REMAINING BALANCE OF LOAN         C
C                  3. RATE    - ANNUAL INTEREST RATE              C
C                  4. TERM    - MONTHS REMAINING OF LOAN          C
C                                                                 C
C EXAMPLE: P&I = $253.83 ON $17000.00 AT 13.000% FOR 120 MONTHS   C
C                                                                 C
C*****************************************************************C
C*****************************************************************C
      SUBROUTINE LOANCALC (OKAY, PAYMENT, LOANBAL, RATE, TERM)
      LOGICAL OKAY
      CHARACTER RESPOND
      DOUBLE PRECISION  PAYMENT, LOANBAL, HOLDPAY, RATE, HOLDRATE
      INTEGER TERM, II
C*****************************************************************C
C                                                                 C
C       START                                                     C
C                                                                 C
C*****************************************************************C
      OKAY = .TRUE.
C******> DETERMINE THE NUMBER OF PARAMETERS SUBMITTED
      II = 0
      IF (PAYMENT.GT.0D0) II = 1
      IF (LOANBAL.GT.0D0) II = II + 1
      IF (RATE.GT.0D0)    II = II + 1
      IF (TERM.GT.0)      II = II + 1
      IF (II.LT.3) GOTO 100
C******> AT LEAST THREE PARAMETERS WERE SUBMITTED
      IF (LOANBAL.EQ.0D0) GOTO 20
      IF (RATE.EQ.0D0)    GOTO 30
      IF (TERM.EQ.0)      GOTO 70
C******> PAYMENT AMOUNT IS EQUAL TO ZERO OR ALL FOUR SUBMITTED
      HOLDRATE = RATE / 12.0D2
      HOLDPAY = LOANBAL /
     +          ((1 - (1 + HOLDRATE) ** (-TERM)) / HOLDRATE)
      HOLDPAY = HOLDPAY + 0.5D-2
C******> IF PAYMENT WAS NOT SUPPLIED, REPLACE
      IF (PAYMENT.EQ.0D0) PAYMENT = HOLDPAY
C******> COMPARE THE SUBMITTED PAYMENT TO CALCULATED PAYMENT
      IF (ABS (PAYMENT - HOLDPAY).GT.6D-2) OKAY = .FALSE.
      IF (OKAY) RETURN
```

```
      DISPLAY "P&I incorrect by more than $.06 ..."
      DISPLAY "Calculated P&I =",PAYMENT
   10 DISPLAY "Continue with corrected P&I (Y/N)"
      ACCEPT RESPOND
      IF (RESPOND.NE."N".AND.RESPOND.NE."Y") GOTO 10
      IF (RESPOND.EQ."Y") PAYMENT = HOLDPAY
      IF (RESPOND.EQ."Y") OKAY = .TRUE.
      RETURN
C******> CALCULATE LOANBAL
   20 HOLDRATE = RATE / 12.0D2
      LOANBAL = PAYMENT *
     +          ((1 - (1 + HOLDRATE) ** (-TERM)) / HOLDRATE)
      RETURN
C******> CALCULATE RATE
   30 RATE = 3.2768D4
      DO 45 II = 14, 0, -1
      HOLDRATE = RATE / 12D5
      HOLDPAY = LOANBAL /
     +          ((1 - (1 + HOLDRATE) ** (-TERM)) / HOLDRATE)
      IF (PAYMENT - HOLDPAY) 35, 50, 40
   35 RATE = RATE - 2 ** II
      GO TO 45
   40 RATE = RATE + 2 ** II
   45 CONTINUE
   50 IF (PAYMENT.GT.HOLDPAY) RATE = RATE + 1.0D0
      RATE = RATE * 1.0D-3
      RETURN
C******> CALCULATE TERM
   70 HOLDRATE = RATE / 12D2
      TERM = 512
      DO 85 II = 8, 0, -1
      HOLDPAY = LOANBAL /
     +          ((1 - (1 + HOLDRATE) ** (-TERM)) / HOLDRATE)
      IF (PAYMENT - HOLDPAY) 75, 90, 80
   75 TERM = TERM + 2 ** II
      GO TO 85
   80 TERM = TERM - 2 ** II
   85 CONTINUE
   90 IF (PAYMENT.LT.HOLDPAY) TERM = TERM + 1
      RETURN
C******> NOT ENOUGH INFORMATION IS GIVEN
  100 OKAY = .FALSE.
      DISPLAY "You did not supply enough information ..."
      RETURN
      END
```

The subroutine above served several purposes. The information concerning potential loans for purchase was stored in an IMAGE database. After the information was entered, we used the LOAN-CALC subroutine to verify that the information was correct. In some cases, one of the components was missing. In cases where information was missing, we flagged the loan and calculated the missing component from the other pieces. When a supplied payment amount was not within an acceptable range, we flagged the loan

and reported the supplied payment amount along with the calculated payment amount for review and correction.

Once the information we were working with was reasonably correct, our next step was to project the loan balances as of some future date when the deal would occur. We took the term we were given and figured the term that would be remaining at some future date. For example, if the term of the loan as of January 1st was 120 months, the remaining term on May 1st would be 116 months. Since we knew the payment amount and interest rate we used LOANCALC to figure the remaining balance on May 1st.

So far, the forecasting is model based. We used the calculations that apply to an individual loan and accumulated the results for the whole group of loans. The underlying assumptions were that everyone would make their payments on time and that no one would sell their house. These were unrealistic, so we had to draw on our past experience with mortgage loans and to review how these loans were expected to perform. Managing delinquent loan projections was made easier by the structure of the deal. The level of default was very low and the towns agreed to assume the responsiblity of replacing or paying for any loans where the individuals ceased to pay. In cases where you need to deal with loan defaults or bad debts, past history is all that you can go on. The history may come from your own operations and experience or from trade association or industry figures. Typically risk is managed by setting aside a reserve for bad debts or defaults, obtaining insurance, and/or decreasing the projected income by some percentage to allow for unanticipated problems. Money is not normally loaned, nor products sold with the expectation of not receiving payment, but in real life it is a part of doing business.

We used a table obtained from a research company to project prepayment of loans. The table showed the survivorship rates for mortgage loans and other types of loans for several public lenders (such as FHA). The values ranged from 100% outstanding at the end of one year to only 13.6% still outstanding at the end of 30 years. What that means is less than 14% of the individuals that borrowed money for 30 year terms paid on the mortgage for the full thirty years. Some may have paid off the loans early, while most moved or refinanced the loans sometime during the thirty years. For our analysis, we applied the figures from the tables to the outstanding loan balances each month. We reduced the outstanding balances by the specified percentages to simulate payoffs. In reality, the payoffs would not be as uniform over the portfolio and over the course of a year. There are normally many less houses sold during the winter than during the rest of the year.

At this point we have simulated the basic loan payment portion of the cash flow. We projected the starting balances and figured the payments each month. We figured the interest earned on reserves at a conservative rate. We allowed for payoffs and prepayment of loans. The other half of the cash flow equation is the outflow to cover the payments on the security.

Securities backed by loans may be structured in a variety of forms from a simple pass-through to a series of multiple securities bearing different rates and maturity dates. Unlike the loans, payments on many notes and bonds are made every six months rather than monthly. You get the use of the money or float during the interim. Other securities may pay interest only at maturity or may be purchased at a discount and redeemed at a higher value at maturity. There is no formula for selecting what form of security to offer. The selection must be based on the variety of securities available to your company, what can be sold to investors, the match with the cash flow from income sources, and a host of other considerations.

TRIAL AND ERROR

I was surprised to find there is no set formula for matching securities to loans. Even the companies that perform cash flow projections for a living use an iterative process to arrive at a workable solution or solutions. This is true for several financial calculations. Look back at the LOANCALC subroutine above. The calculation for figuring the interest rate is iterative. A test rate is inserted, the payment is calculated directly and compared to the actual payment. If we come in high, the rate is reduced. If we come in low, the rate is increased. The process is repeated until we obtain a rate that gives us the payment we specified.

If you doubt this, you should obtain a copy of a manual that comes with a financial calculator (like the HP12C). Look in the manual for the formulas used to calculate the various values. Notice that present value, future value and value at a particular date or term may be specified, but there is no formula for calculating a rate when you know the payment, term amd loan amount. Some manuals even note that an iterative process is used to arrive at an answer. If you calculate bond yields you are also using an iterative process. Bond yield calculations normally assume that you re-invest the interest proceeds at the stated interest rate of the bond. One bond yielding 8% with interest payments every six months will have a different actual rate of return than another bond yielding 8% paying interest at maturity.

NET PRESENT VALUE

Even though the matching of securities to loan proceeds is a trial and error with adjustment process, you will arrive at your result sooner if you start with a good ballpark figure. For a good ballpark figure in our case we calulated the net present value of the loan portfolio. The net present value for a stream of payments is the amount which invested at the specified rate will give you the same result in the end as the stream of payments with re-investment. Another way net present value is described is the value in today's dollars of a stream of payments spread out over time. You may have seen present value used in relation to future value and interest rates. If your goal is to have $500,000 in your retirement account in the year 2020, how much would you need to contribute each month with monthly compounding of interest?

The following subroutine was used to calculate net present value for each loan in the portfolio. The Payment amount and remaining term came from the loan while the rate used was the proposed rate for the security.

```
C*****************************************************************C
C                                                                 C
C                                                                 C
C   NETPRESENTVAL - RETURNS NET PRESENT VALUE OF A SERIES OF      C
C                   PAYMENTS.  PAYMENTS ARE EQUAL OVER THE        C
C                   TERM.  FORMULA IS: SUMMATION FROM I=1 TO N    C
C                   OF THE PAYMENT (I) DIVIDED BY ONE PLUS THE    C
C                   RATE RAISED TO THE POWER OF I                 C
C                                                                 C
C*****************************************************************C
      INTEGER*4 FUNCTION NETPRESENTVAL (RATE, TERM, PAYMENT)
C*****************************************************************C
C                                                                 C
C        RATE    = PERIODIC INTEREST RATE (5% = .050 / 12)        C
C        TERM    = NUMBER OF PERIODS                              C
C        PAYMENT = PAYMENT PER PERIOD (25.17 = 2517)              C
C                                                                 C
C*****************************************************************C
      INTEGER TERM
      INTEGER*4 PAYMENT, PRESENTVAL
      DOUBLE PRECISION RATE, DENOM, RESULT
C*****************************************************************C
C                                                                 C
C        START                                                    C
C                                                                 C
C*****************************************************************C
      PRESENTVAL = 0J
      DO 10 II = 1, TERM
      DENOM = (1D0 + RATE) ** II
      RESULT = PAYMENT / DENOM
      PRESENTVAL = PRESENTVAL + JINT (RESULT + .5D0)
   10 CONTINUE
      NETPRESENTVAL = PRESENTVAL
      RETURN
      END
```

It just so happens that for our case, the net present value of a loan that has the same rate as the rate used for the calculation is equal to the outstanding balance of the loan. If the loan has a higher rate of interest than the security, the net present value is a little higher than the loan balance, while a lower loan rate gives a net present value below the loan balance.

As the final step we added additional components to the forecast to emulate the reserves maintained. We structured the program to accept one or more securities with a variety of starting dates, maturity dates, and interest rates.

To generate the forecast, we ran the program and listed the status of the various accounts at the end of each month. Due to our method of bookkeeping, we had experience with the types of accounts needed to manage loans and securities. By listing the status of each account at the end of a month we were able to tune the forecast and clean up the bugs. We operated on a reduced set of the loans at first and checked the calculations by hand for verification. The program was written to prompt for key parameters so that we could do what-if analysis without recompiling.

The loan data in the IMAGE database was manipulated several ways as a what-if analysis tool. We set up the program to allow selection of loans from all towns or a subset of the towns. We also included some flag or code fields in the database that we could set and select on for obtaining a subset of the loans.

RECOMMENDATIONS

In review, our forecasting proved useful not because it predicted the future, but because it allowed us to assist in answering the question: "What-if?". We purposefully wrote the program and subroutines to be flexible and to allow changes to key parameters without re-compiling. IMAGE allowed us to present variations of the basic loan data to the program. We intentionally avoided the black box syndrome by explaining underlying assumptions and printing the status of each fund account at the each of each month in the forecast. Once the program was de-bugged and stable, we allowed the projection to be generated with less detailed output. One thing we did not do, but which might have been appropriate was to create a sensitivity table incorporating the major what-if components to determine valid combinations. A range of values for each major parameter could be supplied and the computer could crunch on the values generating all possible outcomes that fell within our target range.

ABSTRACT

There is no "EDP user" any more

Walter Willi, BSG Unternehmungsberatung

In the future, there will not only be CIM (Computer Integrated Manufacturing) but also CIV (Computer Integrated Value chain). The complete value chain of a company will be involved in information processing and communications.

The total penetration of companies with EDP and communication equipment will lead to a new division of work between the EDP department and the end user. The bottleneck of implementing an information system are the intellectual capabilities of the end user and not the complexity of the problem to be solved: Rather have a chaos in the computer than in the heads of the user. Thinking in one dimension in information processing is over; we have to accept the "matrix thinking" of the classical organization-approach.

In the future, information processing has to be a plant with its defined duties. Each manager and user has there information system he deservses.

ABSTRACT

Long Range Planning for Now

Paul Wolfe, Missouri Western State College

Missouri Western State Collge has just completed a Long-Range Planning effort for computing and communictions. The approach to planning will be discussed and the successful methods explained. The advantages and disadvantages of committees, task forces, and consultants will be summarized in general and the role of each in the MWSC project will be explained.

Finally, some concrete suggestions will be given as a guideline to mobilizing and sustaining a long-range planning effort.

DATA COMMUNICATIONS
PUTTING THE PIECES TOGETHER

Robert Yori
Hewlett-Packard Co.
3301 Royal Lane
Irving, Texas 75015

Introduction

orks are vastly expanding the uses for computers. Designing a computer network is like putting together the pieces of a jigsaw puzzle.

In this puzzle, what are some of the options for linking not only computers, but terminals, printers, and personal computers into a network? What options are available for

"Putting The Pieces Together"?

Consider the myriad ways in which computer systems and clusters of remote devices can be connected for the purpose of sharing information. Vendors provide hardware and software for connecting not only their systems in a network, but also other computer systems. Some of these communication protocols adhere to international standards, while others are proprietary. Not all vendors offer the same range of capabilities. Even different computers made by the same company may have difficulty communicating.

Exploring different options for connecting systems, terminals, printers, and personal computers stimulates ideas. Let us take a sample manufacturing company with several locations - some near the corporate offices, some in remote cities, and some in other countries.

We can do this in three parts:

1. The Networking Environment

   What is the general business environment, and its associated data processing needs? These answers will help determine the topology of the communications network.

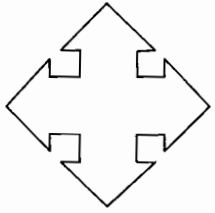
2. Networking Options

   What are some of the different communication protocols, and what are their features?

3. Some Possibilities

   Based upon the options developed in section 2, what are some specific ideas for linking computers and peripherals into networks?

DATA COMMUNICATIONS



Putting the Pieces Together

U1A

The Networking Environment

Consideration of computer networking options within a corporate environment requires that one take into account the business needs of the diverse functional groups.

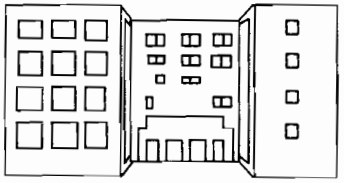
Some of these functional groups might be:

- Administration
- Accounting
- Contract Administration
- Research and Development
- Sales Offices
- Manufacturing
- Personnel
- Shipping and Receiving

Each group has different data processing needs. For example, departments with a a large daily print volume might need a local high-speed printer. This would require a high-throughput communication link to service the printer. Other departments printing at a lower volume would not require such a high-speed, expensive link.

In our hypothetical company, the corporate headquarters are spread among several buildings in a campus setting. The Research and Development department has recently expanded, and is leasing space several blocks from headquarters. Sales offices are located in various cities around the country, with recent overseas expansion.

FUNCTIONAL GROUPS

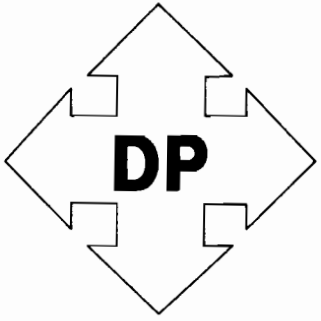


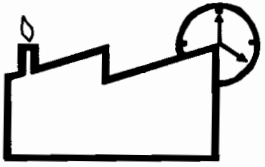
Administration

Shipping & Receiving

Personnel

Accounting



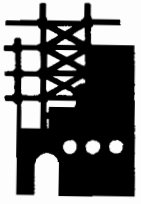


Manufacturing

Contract
Administration

Research & Development

Sales
Offices



U3A

The Networking Environment

Task

To implement a communications network in order to provide computing services to diverse functional groups

Considerations

1. Functionality

   What are the data access and print requirements of each department? Departments such as accounting and contract administration might have high volumes of data input and retrieval. Shipping and receiving's requirements might include such applications as printing picking lists and periodically providing notice of the receipt of shipments.

   A functional network should allow all departments to perform their various tasks in a timely manner.

2. Flexibility

   How rapidly, and how often, does change occur within the company? Do departments move personnel around on short notice? Might additional terminals, printers, or PC's need to be installed in specific work areas in the foreseeable future? Is the company looking to acquire other companies? Is diversification being considered?

   The network should allow for rapid change and expansion, where required.

3. Security

   Access should be on a "need-to-know" basis. The network, along with the corporate computer systems, should ensure that all users have access to their information in a manner consistent with security policies.

4. Cost

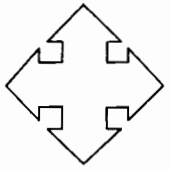
   A well designed network will enable a company to best control communication costs. A network that is poorly designed, whether from the perspective of functionality, flexibility, or security, may result in high or uncontrollable maintenance costs, especially during times of change.

TASK:

Implement a communications network to provide computing services to diverse functional groups.

CONSIDERATIONS:

Cost
Security
Flexibility
Functionality



PUTTING THE PIECES TOGETHER

Hardware Environment

Once the access requirements of the functional groups have been determined, what hardware components will be used within the network? These fall into three main categories.

1. Computers, Terminals, Printers, Personal Computers

   These devices would most likely be installed throughout the company. The computer systems may be located in various departments, or even in different computer rooms in the same facility. Often, the hardware is from different vendors.

2. Communications Links

   Different types of cables can be used to connect systems, terminals, and PC's into a network.

   Coaxial cable is used for Ethernet and IEEE802.3 local area networks. IBM's System Network Architecture (SNA) uses coaxial cable on some terminal links, and RS232-type cables or V.35 cables on system-to-system links.

   Some communication methods can be used interchangeably in some networks, such as telephone company leased lines or microwave links. Costs may vary considerably. With a good business plan, the limits of spending for the network links will already have been determined.

3. Data Communications Equipment

   For connecting computer systems, terminals, printers, and PC's, we can use data switches, private branch exchange's (PBX's), local area networks (LAN's), or multiplexor's (MUX's) Variables include line speed, throughput, installation costs, and network maintenance costs.

   To determine where breaks have occurred in communication circuits, devices such as:

   protocol analyzers
   intelligent modems - for automatic testing
   local area network analyzers

   are needed. In conjunction with these diagnostic devices, we will need good network trouble- shooting software.

HARDWARE ENVIRONMENT:

1. Computers/Terminals/Printers/PC's
   - Multiple locations
   - Various capabilities
   - Different vendors
2. Communication links
   - Local: 4-wire / COAX / fiber optics
   - Telco: analog / digital / satellite
3. Communication equipment
   - PBX's/Data Switches/LAN's
   - Multiplexors and modems
   - Test equipment

Networking Options

Given that the various departments within our company have different computing needs, different requirements for portability of devices, and different requirements for data throughput, the network that best serves these departments will most likely be a hybrid network, using various communications methods to link user groups to the corporate computer systems.

In general, networks can be classified by architecture, falling into 4 main categories.

1. Hierarchical

   Here, a host system controls and directs all the "data traffic" in the network. IBM's SNA architecture is an example of a hierarchical network.

2. Backbone

   These networks use a coaxial or fiber optic cable as the main trunk line connecting systems, groups of terminals, and personal computers. Ethernet and IEEE802.3 local area networks are backbone networks. Wide area networks such as community cable TV systems are also backbone networks.

3. Point-To-Point

   These are the simplest networks. Punch card or key-to-disc remote job entry (RJE) systems are prime examples.

4. Hybrids

   In these networks various technologies are merged. Thus they warrant more detailed examination.

NETWORK OPTIONS:

Several architectures usually must be merged to provide computing services to all functional groups.

NETWORK ARCHITECTURES:

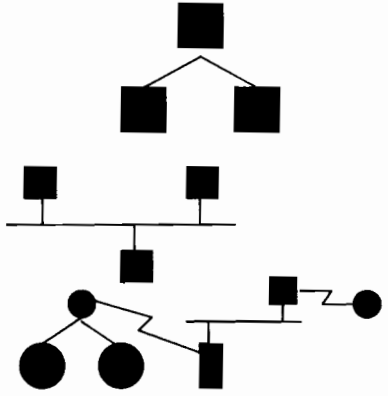
Hierarchical

Backbone

- PC Networks
- Local Area Networks
- Wide Area Networks

Point–to–Point

Hybrids



Strategic Network Planning Issues

In developing a hybrid network, different connection methods, with varying protocols, must all be merged so that information passes between many points efficiently, and without error.

Information may come in from a remote branch office via a public X.25 network and be printed in the accounting office, where the printer is attached to the the central computer system via a local area network.

By paying attention to the following issues during the network planning phase, a hybrid network can be developed that passes information between users and systems in a "seamless" manner.

1. Standards

   Use of network protocols that adhere to standards greatly reduces the risk of network obsolesence.

   Some international standards are:

      IEEE 802.3/4/5 - for local and metropolitan networks
      X.25 - for wide area networks

   Some defacto standards are:

      SNA - for IBM communications
      RJE - for system-to-system file transfer

   One point of caution. Vendors may implement standards differently, producing systems that cannot pass data to each other, even though they are using the same communications protocol.

(continued)

2. Compatibility

If a company's computer systems are made by one vendor, in most cases it will be easiest to use that vendor's communications products, not only to enhance compatibility, but also to ensure better support. Here we assume that the products have the functionality desired.

Some emerging standards that show promise as ways of linking dissimilar computer systems are:

A. PU2.1/LU6.2 & APPC

Physical Unit 2.1 and Logical Unit 6.2 are relatively new definitions within IBM's SNA architecture. Advanced Program To Program Communications (APPC) is the set of software routines used by programs communicating via PU2.1/LU6.2. Together, they permit peer-to-peer communications between systems in an SNA environment, which has traditionally been a hierarchical communications environment.

Since all major vendors offer products for communicating in the SNA environment, PU2.1/LU6.2 offers a method whereby dissimilar systems may possibly communicate, either for the purpose of passing files, or in a program-to-program (real time) environment.

B. X.25

This protocol has been successfully implemented for world-wide communications by both private corporate networks and public network companies.

Multiple, dissimilar systems, communicating via various communication methods, can all be blended onto one network. Communication lines can easily be shared, hence the use of this protocol by public network companies.

Most major vendors offer some form of X.25 communications on their systems. More vendors communicate via X.25 than currently with PU2.1/LU6.2 simply because X.25 as a standard has been implemented for a greater time period.

(continued)

3. Network Redundancy

   If alternate communication paths around critical points in a network are planned for from the beginning, then the vulnerability of systems and terminals to network malfunctions will be greatly reduced.

   SNA and X.25 are two of the more popular communication methods that have redundancy capabilities built into the software protocols.

4. Central Network Error Detection

   National or international networks may have many systems and groups of terminals attached. If a problem does occur at some location, it would be best to have that problem reported to a central technical staff.

   This is a function of the network software, and the hardware devices controlling the network.

   Some form of central control will be required for any large network.

Strategic Network Planning Issues

Standards

- International — IEEE 802.3/4/5 & X.25
- Defacto — SDLC/SNA & BSC/RJE

Compatibility

- Between vendors — RJE, LU6.2/APPC, X.25
- Within a vendor's line — SNA, NS & others

Network redundancy — X.25 & SNA

Centralized trouble-shooting

Local Area Networks

Ethernet and IEEE 802.3 are two communication protocols that define procedures for communicating between systems and clusters of terminals/PC's via a coaxial backbone cable. Various types of coaxial cable can be used.

Both of these protocols use the same system and network hardware. However, the actual block structures of the packets that are sent between systems differ.

Ethernet systems and those using IEEE 802.3 protocols can co-exist on the same coaxial cable link, but they cannot communicate with each other, due to this difference in the block structures.

Advantages of these protocols are:

1. There is no hub system on the network. Any system can be "down", and the others can still communicate.

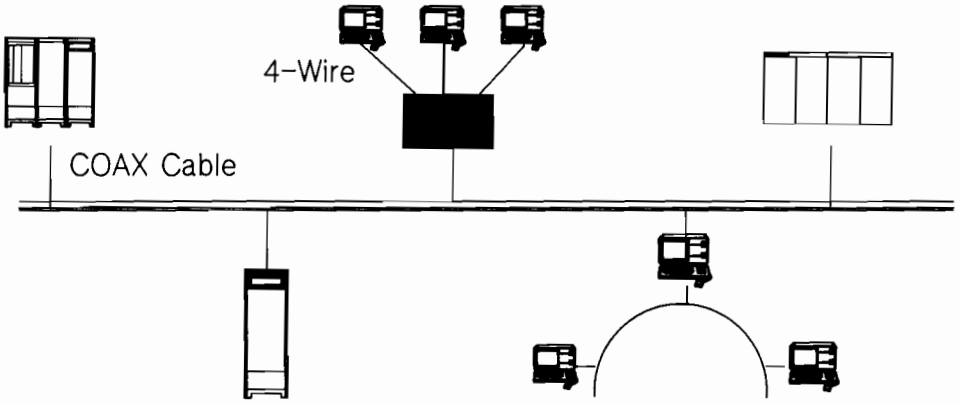
2. The link runs at 10 megabits/sec (1.25 million char/sec). Large file transfers are now possible on a frequent basis between systems.

3. The protocols allow for the coaxial cable to be up to 500 meters (approx 1600 ft.) in length, with a maximum of 5 cable segments joined by electrical repeaters.

   Be aware of the "2 repeater rule". The data path between any two computer systems cannot cross more than 2 repeaters. This means that the actual maximum distance between any two systems that must communicate is 1500 meters (approx. 4800 ft.).

IEEE 802.3 & ETHERNET *

CSMA/CD



High connectivity – no HUB system

High speed communications – 10 megabits

Local environment – maximum 2500 meters

Multi-vendor communication

*** Can co-exist**

Two other protocols defined by the IEEE committee are:

1. IEEE 802.4

   This protocol is most commonly used in community cable TV networks. Coaxial cable is used as the backbone, and has a transfer rate of 50 megabits/sec (5 times faster than IEEE 802.3).

   One advantage of this protocol is that data and voice can be carried simultaneously on the network.

2. IEEE 802.5

   This protocol is implemented by IBM for the PC token-ring network.

   IBM's implementation, a superset of the 802.5 protocols, defines a 4 megabit transfer rate.

   These token-ring networks also interface to SNA networks via PU2.1/LU6.2.

IEEE 802.4

- Wide-area network (WAN)
- 50 megabit transfer rate
- GM/MAP

IEEE 802.5

- Token-ring network
- IBM implementation a superset of 802.5
  - LAN for personal computers
  - 4 megabit transfer rate
  - SNA interface via LU6.2

U11Z

X.25 Private Data Networks

As mentioned earlier, the X.25 protocols allow multiple systems and clusters of terminals to share communication lines. Hardware devices called X.25 switches make this possible. Redundant communication routes can also be designed into the network.

Some advantages to an X.25 network are:

1. High connectivity can be designed into the network. There is no hub system in the network. Like the IEEE 802.3 LAN's, any system can be down, and the other systems can still communicate.

2. Line speeds vary from 2400 bits/sec to 56000 bits/sec. These are telephone line speed constraints.

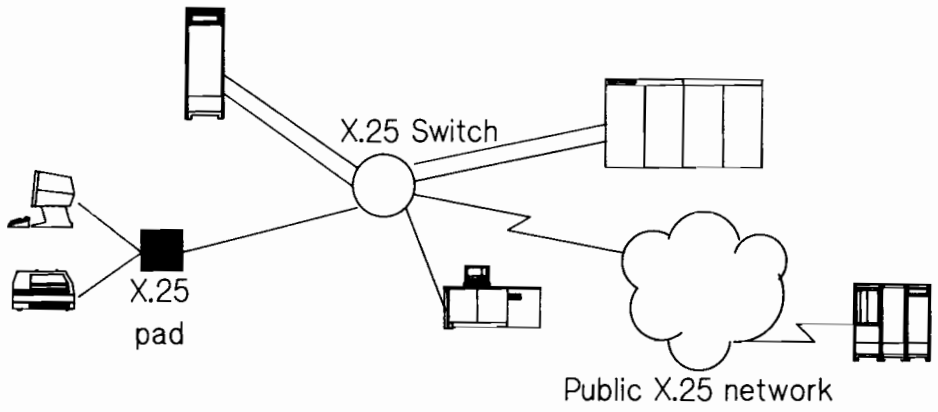
3. Multi-vendor communication is possible. For example, communication is possible between IBM mainframes and HP systems via an X.25 network.

4. These networks can be used for world-wide communications, or the linking of systems within the same computer facility. This network flexibility was one of the main considerations during the initial design of the network.

5. Concurrrent dissimilar data streams are possible. For example, two systems can transfer data via RJE, and two other systems can communicate via SNA, and all share the same communication line. In these cases, the RJE and SNA data blocks are re-packaged into X.25 packets, sent over the network, and then broken out into their appropriate formats.

6. Public X.25 networks throughout the world have gateway links, so that all these networks interface. The end result is that data can literally be transferred around the world.

X.25 PRIVATE DATA NETWORKS



High connectivity – no HUB system

Medium speed – 4800 to 64000 bits/sec

Multi–vendor communications

Concurrent dissimiliar data streams

Local or world–wide communications

U12A

SNA - System Network Architecture

SNA has traditionally been a hierarchical protocol, with an IBM 370-compatible mainframe acting as the host. IBM 370 architecture has been implemented in all host machines from the original 370's up to the current 309x Sierra series.

SNA categorizes hardware devices into 5 types or Physical Units (PU's).

PU 5 - host system
PU 4 - communications front-end processor (3705/3725)
PU 3 - not defined
PU 2 - terminal cluster controller or distributed system such as System 36's, System 38's, Series 1's, or 8100's.
PU 1 - terminal controllers upgraded to run in an SNA environment

Associated with each hardware device is Logical Unit (LU) software. For example, a PU2 printer has associated with it a different LU than a PU2 terminal.

With the new PU2.1/LU6.2 definitions, peer-to-peer communications is supported between PU2 devices (except terminal controllers). Now, System 36 and System 38 computers can transfer data, without that data having to be routed through the PU4 devices (front-end processors). Also, PC's can communicate directly with systems via PU2.1/LU6.2.

Some advantages to SNA are:

1. Line speeds range from 2400 bits/sec to 56000 bits/sec. Between PU5 devices, the data runs at channel speeds (10 megabits/sec).

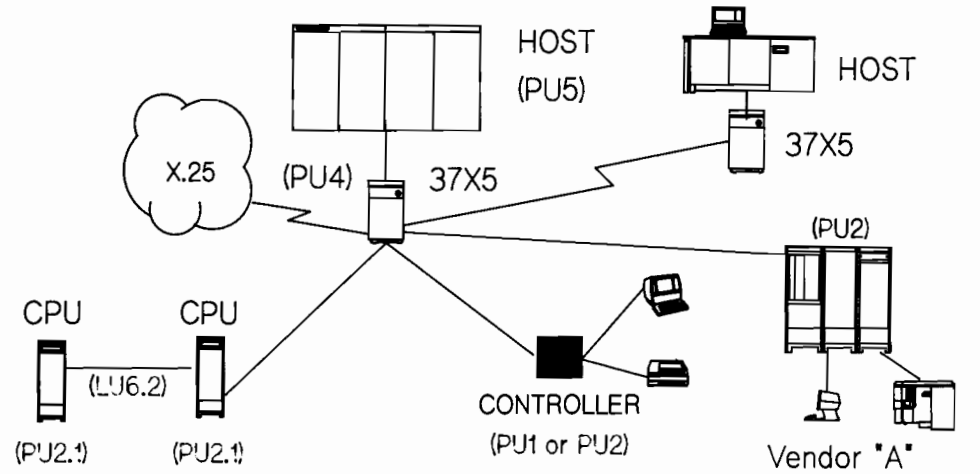
2. Useful for local and world-wide communications.

3. Peer-to-peer communications is possible between traditional PU2 devices when using the PU2.1/LU6.2 protocols.

4. Interface to X.25 networks is possible with the software package NCP/PSI running on the front-end processor.

5. Redundant communication paths can be defined between front-end processors.

SYSTEM NETWORK ARCHITECTURE



Hierarchical

Medium speed

Local and world-wide communications

Peer-to-peer with PU2.1/LU6.2

X.25 links with NCP/PSI

U13A

Networking Options

Remote Job Entry

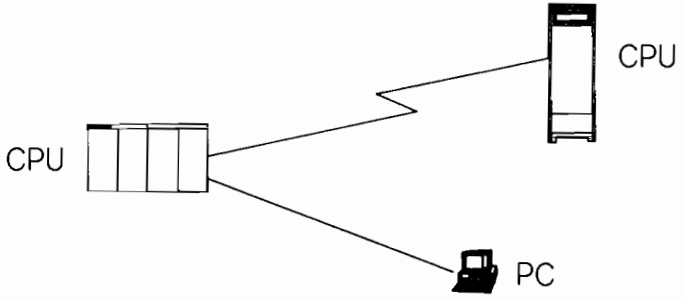
RJE is a communications protocol that was developed in the early 1960's. It is a defacto communications standard.

Almost every computer vendor offers an RJE package on its systems, and even on its PC's.

Due to the availability of this software on most systems, it may be the easiest (or only) way to transfer data between systems, especially if these systems are from many different vendors, and of various "vintages".

This is a batch data transfer protocol, with line speeds running from 2400 bits/sec to 56000 bits/sec.

REMOTE JOB ENTRY



Multi-vendor communications

Medium speed

Batch only

At times the ONLY way to communicate

U14A

Networking Options

PBX's - Private Branch Exchanges

Most large and medium companies are installing PBX's to control their internal telephone network. Modern PBX's allow for the transmission of voice and data over the same set of telephone wires. With this capability, plus the appropriate computer-interface hardware, a terminal can literally be installed anywhere there is a telephone.

PBX's also allow dial modems to be shared. A pool of modems can be defined so that any terminal user on the PBX can access them.

PBX's can be connected in various ways using telephone lines or fiber optic cables. This allows the phone systems of several departments to be merged into one telephone network. This would reduce the number of long distance calls between offices located in different cities.

Depending on the vendor, some PBX's will interface with PC networks.

Some advantages of PBX's are:

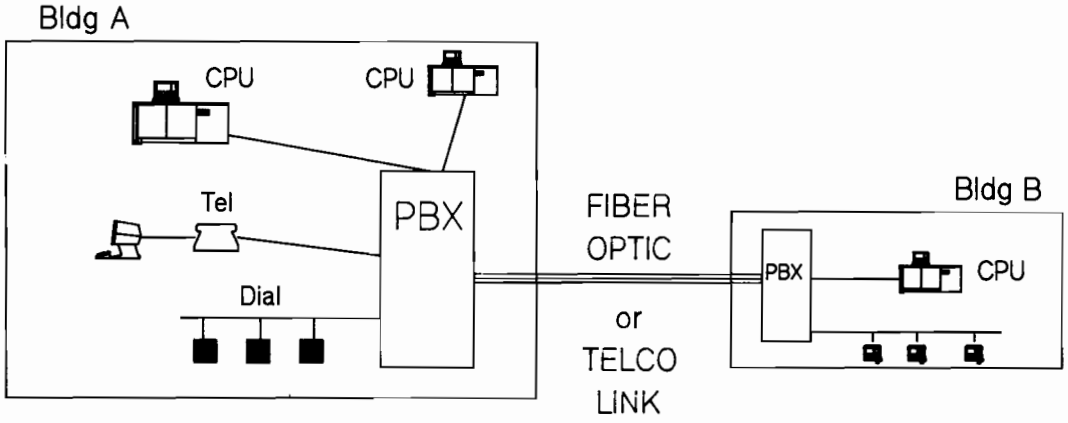
1. High connectivity - terminal at any telephone

2. PBX can act as a data switch between multiple systems

3. High terminal speeds - up to 19,200 bits/sec

4. One-time wiring and installation costs.

PBX



High connectivity – terminal at any tel.

PBX acts as switch for terminals to CPU's

High terminal speeds – up to 19.2 kb

One–time wiring installation & cost

U15A

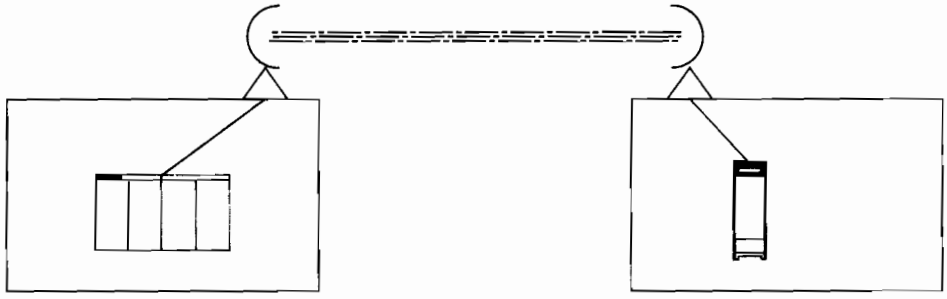
Microwave Links

Microwave links can be used for communicating between computer systems, or a system and a cluster of terminals.

Approval by the Federal Communications Commission (FCC) is required, and can be the most lengthy part of the entire installation.

Line-of-sight clearance is required between the microwave transmission/receiving devices. Also, transmission can be impaired during heavy rains, or in fog.

MICROWAVE



FCC approval required

Line-of-sight / parking lots

Medium distances

CPU - CPU or CPU - terminal connections

U16A

Multiplexors and Modems

At times there is a requirement to provide access to central computers for a remote cluster of terminals, PC's, or printers.

The most expensive, and least desirable way to do this is to provide either individual dial-in access, or separate telephone leased lines for each device.

Since most remote terminals and printers are not active all the time, these devices can share a communications line. A MULTIPLEXOR makes this possible by taking the information for/from remote devices, packing it in blocks, and sending the blocks over the communications line.

Line contention is a variable. A 300 line-per-minute printer can take the entire band-width of a 9600 bits/sec communication line while printing. The same is true when many active terminals are running at high speeds on the line.

MODEMS (modulator/demodulator) take the digital signals from terminals, systems, etc., and convert them to analog signals, allowing transmission over the telephone network.

Each multiplexor requires a modem. So, for a remote cluster of terminals, 2 multiplexors and 2 modems are required, along with the communications link.

For digital telephone circuits, modems are replaced with Digital Service Units (DSU's).

The following are factors we must consider when using multiplexors in a network.

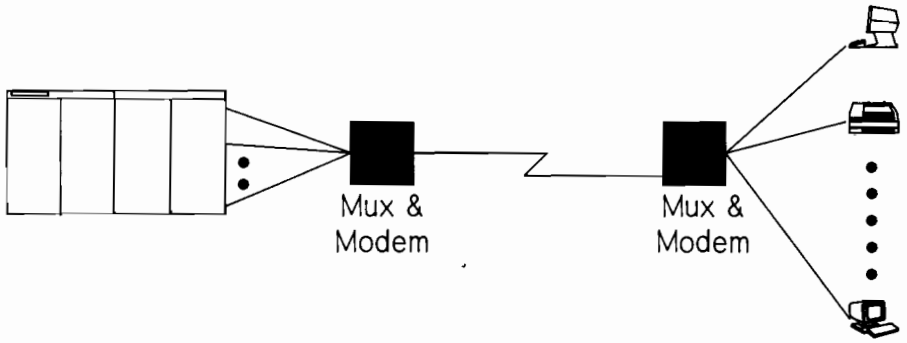
1. Line speeds are limited to 19,200 bits/sec.

2. Choose multiplexor/modem combinations preferably from the same vendor.

3. Select multiplexors and modems for their remote diagnostic capabilities.

4. Vendor's hardware support options range from mail-in service to on-site service. Note that some only offer mail-in hardware service.

MULTIPLEXORS & MODEMS



Line speeds up to 19.2 kb
Line contention a consideration
Choose MUX/MODEM for remote diagnostic options
Design network using one vendor's equipment
Consider hardware support options

Corporate Services

Departments such as administration, accounting, contract administration, and personnel can be considered as part of corporate services.

These departments usually have large staffs. Some personnel are assigned full-time to data entry tasks, while others only casually access the corporate systems. Some users may require access to many different computer systems in the network.

One way to solve the information access needs of these departments is to use the existing company PBX. By adding data modules to existing telephones, we can install a terminal at any desk.

With the high terminal speeds of 19,200 bits/sec supported by the PBX, data entry/retrieval will be fast.

As personnel move, no additional wiring will be required. Simply move the terminal and the telephone data module.

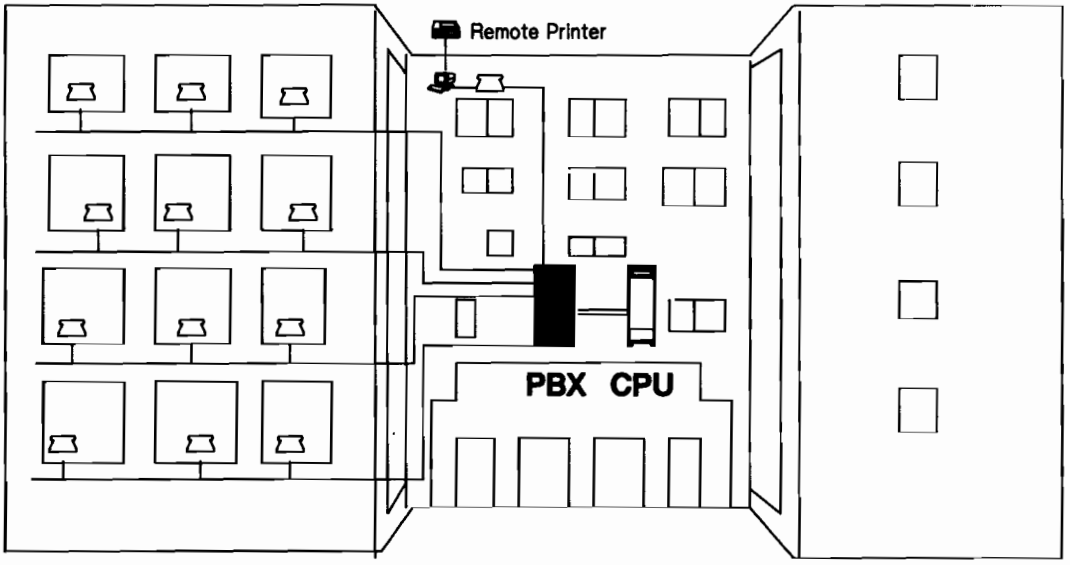
Access to any of the computer systems linked to the PBX can be controlled. Managers needing access to data bases on several systems can be allowed security clearance. Other users, such as data entry personnel, can be granted access only to the systems on which they need to work.

Access to other systems in the network that are not attached to the PBX must be handled differently. One of the computer systems attached to the PBX could act as a "gateway" system to the rest of the network.

With the number of terminals that can be attached to the central systems through the PBX, data can be captured at its source. Users from the individual departments have a sense of ownership of the data, thereby improving the accuracy of the data.

We could also install low speed remote printers in work centers in the various departments. The printers are attached to the system in the same manner as terminals or PC's.

Corporate Services



Use PBX for access to central system from many departments

Improved accuracy:

- **capture data at the source**
- **users have control/ownership of input and reports**

U19A

Manufacturing Facility

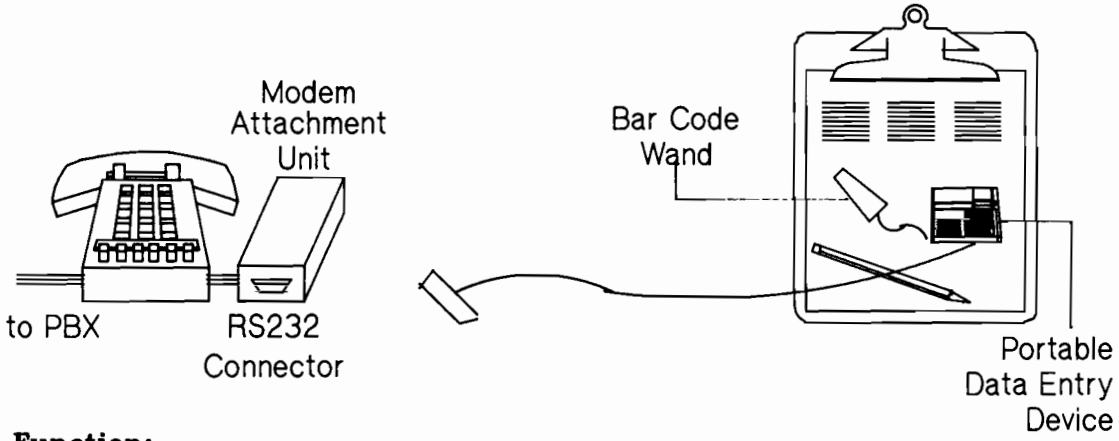
Here we can extend the use of the PBX, as discussed for the corporate services departments.

In a manufacturing facility, the movement of parts, and their changing status, may need to be logged at various points in the manufacturing process.

By using portable data collection devices, possibly with bar code readers attached, information could be captured concerning the status of a part during any phase of the manufacturing process.

By using portable data collection devices with RS232 interfaces (like those on terminals) AND by providing attachment points into the computer system at the various logging points (like the telephone connections to the PBX), data can be captured in an on-line fashion and sent directly to the main computer facility.

Manufacturing Facility



Function:

1. **Connect to modem**
2. **Enter part number using bar code wand**
3. **Select activity on portable**
4. **Enter information**

Inventory part
with bar code ID

U19Z

Sales Offices

If we have only several remote sales offices, multiplexors connected to central computers via leased lines might be the most cost-effective networking option.

If there are offices in many cities, states, and countries, then a private or public X.25 network may the best way to interface remote multiplexors.

The trade-offs between using a public X.25 network or setting up a corporate private X.25 network are mainly in the areas of cost of the network and net throughput of data.

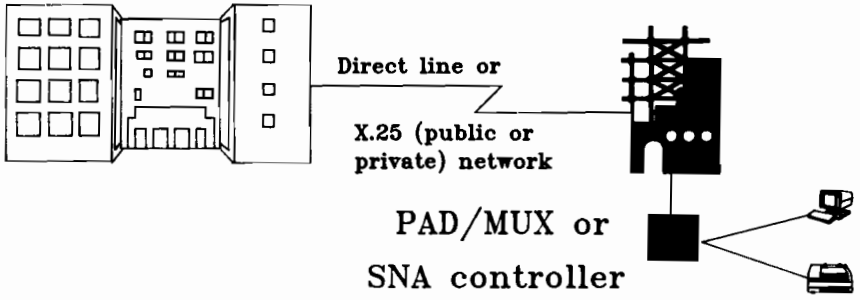
Depending upon the country and the tariff regulations, public networks may be less expensive than private networks. This is the case in Europe. In the United States, the situation is reversed - private networks are less expensive in high-use situations.

In either case, a well-designed private X.25 network will provide better data through-put than a public network. In a public network, the user has no control of network load; the more users on the network, the longer it takes data to get to its destination.

Public networks are designed with redundant communication paths, and provide network trouble-shooting as a part of their service. With a private network, our company must establish redundant data paths and maintain a staff of network technicians.

A combination of public and private X.25 networks might be appropriate. For those sales offices with little daily volume, either in data entry/retrieval or printing, or for those sales offices in other countries (Singapore, for example), access could be provided to the computer systems by way of a public X.25 network. The other high-volume sales offices could be linked through a private X.25 network.

Sales Offices



Access to power of corporate systems

Remote data entry & inquiry–
better coordination of information

Communication costs may vary

U20A

Some Possibilities

Administration and Accounting

These departments could gain access to the corporate computer systems through the PBX, as discussed previously.

An alternate method would be to use a local area network.

Multiple PC networks can be easily integrated with a LAN. providing for high speed transfers between PC's and peripherals attached to PC servers.

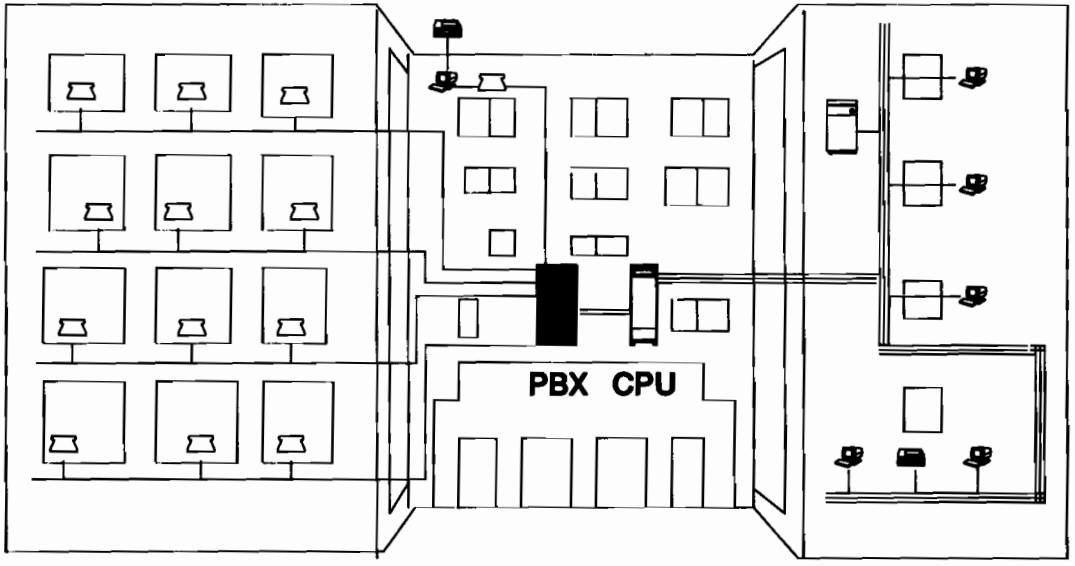
Mini-computers, acting as office automation systems, can also be integrated with the PC networks. Here, the computers function as an alternate server for any of the PC networks.

All computers attached to the LAN can share data either between themselves, or with the PC networks.

Medium and high-volume printing can be provided for the accounting staff and corporate officers by way of laser printers attached to the LAN, with a transfer rate of 10 megabits/sec.

If users from departments attached via the PBX to the corporate systems need to access the LAN, one of the central systems would act as a gateway.

Administration / Accounting



Local or wide area network (LAN/WAN)
Low to medium printing
High–volume inquiry and data entry
Departmental office automation CPU's

U21A

Some Possibilities

Research and Development

As mentioned in the initial description, the Research and Development (R&D) department recently moved to a facility several blocks from the corporate headquarters.

Access for terminals, printers, PC's, and even systems in the R&D department could be provided by leased lines, using several of the technologies we have discussed, such as SNA or X.25.

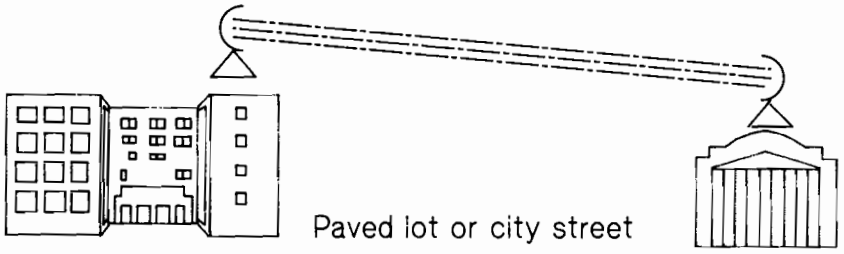
If a line-of-sight can be established between the corporate computer center and the R&D department, then an alternate method to leased lines would be a microwave link.

Microwave transmission runs at speeds of 1.544 megabits/sec (T1 speeds). This transmission "band width" can be separated into channels, so that systems and terminal/PC clusters can share the microwave link.

One advantage of establishing a microwave link is that the costs are fixed - there are only purchase, installation, and maintenance costs.

In campus situations, where a parking lot or field separates buildings, a microwave link can be established without digging or tunneling, as would be required for a modem link or a LAN link.

Research & Development



Microwave link (station KBUS)

One-time installation cost

No leased-line charges

No destruction of existing surfaces

U22A

Some Possibilities

Data Processing

In the corporate data processing center, the systems can be linked by a variety of methods.

The slide shows both PBX, LAN, and X.25 links between the computer systems. While all three of these would not necessarily be installed concurrently, what we see here is the flexibility and redundancy provided by combinations of these communication methods.

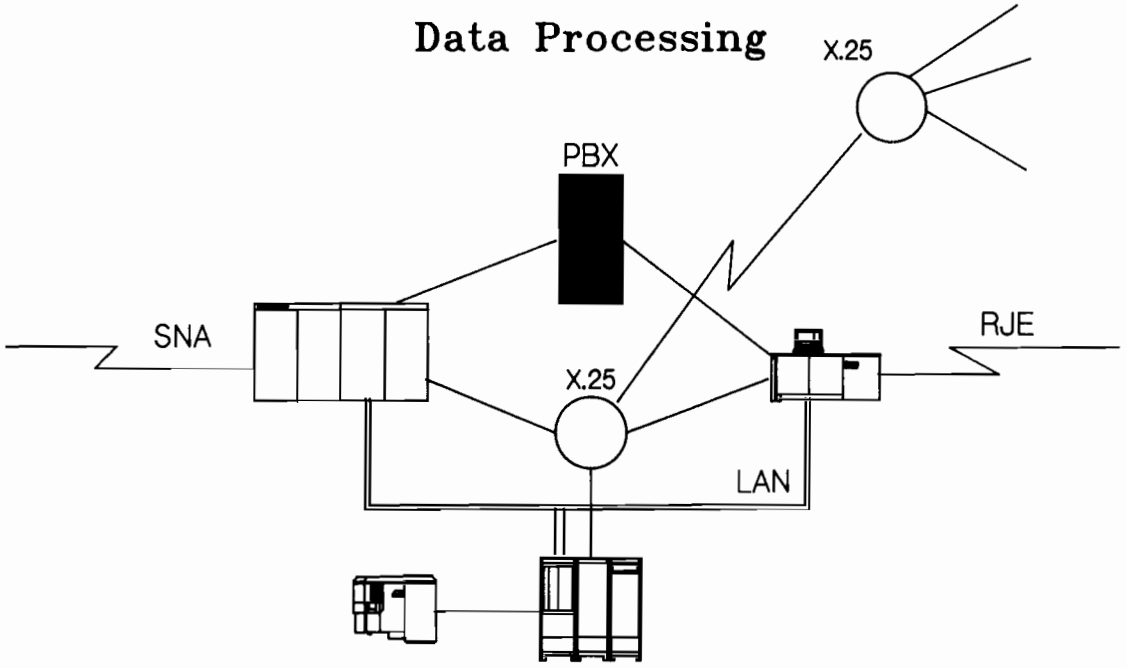
Systems in the network can function as "gateways" to other communication environments, such as SNA or bysync (with RJE).

High speed, expensive peripherals such as laser printers can be accessed by many systems. Here, the LAN interface provides the optimal solution, since it offers the highest data transfer rate - 10 megabits/sec vs the 56000 bits/sec of X.25.

The LAN link can be used to access shared disc drives. X.25 could also be an option, but the high data transfer rates of the LAN make it the connection method of choice.

Both the X.25 and LAN links allow data bases to be shared by the applications on all systems in the network.

Data Processing



High connectivity (X.25 or LAN)
Share high-speed printers
Share data bases
Medium/high data transfer requirements

U24A

Conclusions

Any well-designed computer network must meet the following criteria:

- Functionality
- Flexibility
- Security
- Cost Control

It must also take into consideration the adherence of the communication protocols to standards.

The various communication methods (SNA, X.25, LAN, and others) all meet these requirements to varying degrees.

Concerning costs, SNA may be the most expensive network to install because of the costs of the host computers and the front-end processors. That is, unless systems are linked using PU2.1/LU6.2, in which case no host system is required.

Large X.25 networks, especially those with many redundant communication paths, can be expensive because of the costs of the X.25 switches. But, line costs will usually be less than those of a large network of multiplexors, where separate leased lines run from central computers to remote offices.

For both SNA and X.25 networks, network trouble-shooting software and hardware can be expensive - but it is well worth the investment.

PBX systems vary in cost, depending on the number of telephone nodes to be installed.

(continued)

Conclusions

For local and wide area networks, the coaxial cable purchase and installation costs can be a major element in the total cost of the network. The exception is a LAN installation where all the computers are in the same room.

The more initial network analysis and planning done, the more successful the network installation will be. This analysis will provide for not only the

- Functionality
- Flexibility
- Security
- Cost Control

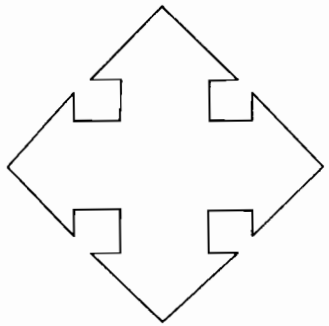
required, but define the potential for growth in the network.

The network analysis, reflecting the objectives of the corporate business plan, will aid the corporation in

"Putting The Pieces Together".

Cost

Functionality



Flexibility

Security

U25A

Computer Crime - What To Do Before It Happens

Bernard P. Zajac, Jr.
Database/Data Security Manager
ABEX Railroad Products Group
200 South Michigan Avenue
Chicago, Illinois 60604
(312) 322-0360 Telex: 756108

ABSTRACT: Many computer installations do not have any formal plans on what to do when it has been a victim of a computer abuse. The concept of the Computer Abuse Response Team is proposed, and ways of preserving evidence to make prosecution easier are presented.

Presented to the INTEREX 87
September 1987
Las Vegas, Nevada

An early version of this paper was presented to Securicom'86 in Paris, France, March 1986.

ACKNOWLEDGMENTS

The author would like to thank the following people for their comments on the early drafts of this paper: Mr. Tom Ryan - ABEX Corporation; Mr. Kirk Tabbey, Esq. - Washtenaw County Prosecutor's Office; Mr. Robert Brown, Esq. - Schlussel, Lifton, Simon, Rands, Galvin, Jackier; especially, Mr. Kevin Morison - Illinois Criminal Justice Information Authority.

I. INTRODUCTION

It's late. You are the data processing manager for a large east coast bank. You're watching the 11 p.m. news when you get a call from your night operations supervisor, who says he believes that his staff has just aborted an unauthorized remote terminal session. What do you do?

Or, the FBI walks in with the following printout:

Numb: 212
Sub: BANK SYSTEM
From: THE SERPENT (#76)
To: ALL
Date: 4/14/85 11:55:40 AM

This number is to a BANK system, believe it's a DEC:

212-555-5000

LOGIN 1,4 pw:SECRET

USE AND ABUSE!!!!!!!

The FBI informs you that the telephone number is to your bank, and it has been posted on a computer bulletin board for at least a month. Again, what do you do?

These scenarios are happening more often than people want to admit. This paper discusses not so much how to prevent these situations, but rather what should be done when a problem is uncovered.

How Much Computer Crime Is There?

Joseph O'Donoghue of the Department of Behavioral Science at Mercy College in Dobbs Ferry, New York, recently surveyed the chief executive officers of the Forbes 500 corporations concerning computer crime within their organizations.1 O'Donogue found that of the 184 respondents surveyed, 56 percent had experienced a computer abuse incident, and the average loss was $118,932.2 Roughly 79 percent of the incidents were committed by individuals that had contact with the organizations employees, contractors, competitors, or customers.3

While the dollar amount per incident is lower in O'Donoghue's report, it tends to support the American Bar Association's (ABA) 1984, report, "Report On Computer Crime." The ABA surveyed selected firms from Fortune magazine's list of the 500 largest U.S. corporations, all major federal departments and agencies, the attorneys general of all 50 states, and some local prosecutors.

The ABA found the average financial loss per incident

1Joseph O'Donoghue, "Strategies Found To Be Effective In The Control of Computer Crime In The Forbes 500 Corporations," SIG Security Audit & Control Review, ACM, New York, N.Y., p 1.

2Ibid., p. 2

3Ibid., p. 3.

ranged from $2 million to $10 million.4 While personal financial gain was the most cited reason for computer abuse, "intellectual challenge" was the second most common factor.5 The ABA noted that, while the bulk of the abuse had been perpetrated by individuals from within an organization (77 percent), the abuse committed by people on the outside had been done by individuals who never had any prior contact with the organization before the abuse.6 O'Donoghue had similar findings.7

How The Police And Court Response Has Changed

Since the first documented computer crime case in 1958,8 police, prosecutors, and businesses have changed their response to the problem.

In investigating the early computer crime cases, the police, who generally lacked expertise in data processing, found themselves hampered because computer crimes were different from traditional crimes. With computer crimes, a person could "take" something, but that something would still remain in the system, and no physical evidence existed, in the

4American Bar Association, "Report On Computer Crime," Washington, DC., June 1984, p.15.
5Ibid., p. 22.
6Ibid., p. 19.
7O'Donogue, pp. 11-12.
8Donn B. Parker, Crime By Computer, New York, NY: Charles Scribners & Sons, 1976, p. 34.

traditional sense, to show that something had been stolen. Furthermore, many computer systems lacked audit controls and/or trails to show that something was amiss.

James A. Sprowl, professor at the Illinois Institute of Technology, Chicago-Kent College of Law and noted author in the field of computer-generated evidence, underscores the importance of proper audit trails. Sprowl notes, "Situations where you really have difficulty with a computer crime and its proof are really situations where the computer system was not designed with the proper controls in place to either prevent problems or at least make an audit trail of their occurrence."[9]

Some problems during the early investigations of computer abuse stemmed from the laws that defined criminal acts. Those laws did not cover computers per se, thus forcing prosecutors to use existing laws, many of which dated back to English common law. Since those laws did not specifically mention computers or computer software, evidentiary problems arose. How do you present a reel of computer tape, or a disk containing a program or data, and prove that it is worth more than the actual cost of the tape itself? Admissibility problems arose when prosecutors attempted to introduce computer-generated reports as evidence. The introduction of the original records

---

9Bernard P. Zajac, Jr.,"Police Response to Computer Crime", The Computer Law and Security Report, July/August 1985.

could be impractical or cost prohibitive. In some instances, such as internal audit trails, the internal audit trails themselves ARE the records of original entry created by the computer.

Because of these problems, it is very important to deal with computer abuses in a proper manner to insure the admissibility of evidence, both computer-generated and non-computer-generated.

II. What To Do Before An Abuse Happens

Once it has been discovered that a computer system has been compromised, several things must happen immediately:

- The intrusion stopped
- The loss estimated
- A determination of what, if any, action should be taken. The first action to take when a compromise has been discovered is to stop it. A number of abuses occur through the use of a valid user ID and a dial-up. However, this is one of the easiest to halt once it has been discovered. The circuit can be disconnected and the ID invalidated. The holder of the id should be required to meet with the data processing manager before the ID is reactivated. This allows the data processing manager to find out if the ID was lent to an unauthorized user and to restate corporate security policy. Should the abuser <u>not</u> be known right away, it may be necessary set a trap, let him or her back into the system, and have the phone company traces the circuit.

Since many state laws have a "financial trigger" determining whether or not the crime is a felony or a misdemeanor, the loss must be estimated so a determination can be made whether or not prosecution is warranted, This decision will have a bearing on how to the case is going to be handled.

The Computer Abuse Response Team

Ideally, an installation should have procedures developed and a Computer Abuse Response Team in place long before a computer abuse ever occurs. This Computer Abuse Response Team should include the the data processing manager, the data security manager, the corporation counsel, the controller, and, if the installation has one, the data security manager. Ideally, the data security manager should head the team. Should the installation not have such a manager, the data processing manager should be designated the leader. The team can be expanded, on an as-needed basis, to included telecommunications personnel, technical support, or EDP auditors.

Each team member has been chosen because of his or her position in the organization:

The data processing manager - the person who controls and is ultimately responsible for the computer center. He or she will be able to give the team insight on what the impact of

an abuse would be at the installation.

The data security manager - the person who is responsible for the security of the company's data and is familiar with computer investigation and crime. This individual should head the team and, working with the director of security, act as the team's liaison with the police.

The corporation counsel - the person who will aid the team in the legal complexities of evidence, protect the company from possible suits that might result from any disciplinary actions, and work as a liaison with the data security manager to the prosecutor's office.

The controller - the person who as chief fiscal officer, will be able to give insight into the financial implications of the abuse and the financial ramifications of prosecution.

The director of security - the person who as head of security, will bring special skills in interviewing suspects, provide logs of personnel activity in restricted areas such as tape vaults, provide assistance in securing a place for evidence, and probably an ongoing dialogue with the local police.

When an abuse is discovered, the Computer Abuse Response Team is activated. Once activated, it should immediately assess the extent of the loss or damage and then turn to the senior management team for direction on whether or not to

prosecute.[10] The controller and data processing manager should work closely in estimating the monetary value of the abuse.

Should the case not be prosecuted, but handled internally, the response team's legal member should still play an important role in the team's actions, especially in protecting the company from any suits that my result from any disciplinary actions. In addition, the company should execute a non-disclosure agreement with the responsible parties so they will not divulge any information gained or tell anyone else how to compromise the system.

If the case is going to be prosecuted, it is extremely important to gather evidence correctly. Successful prosecution depends on the ability to <u>prove</u> that a crime has occurred, and this requires evidence.

[10]It should be pointed out that certain crimes MUST be reported to the proper authorities or the company may be liable.

Evidence

When using computer-generated evidence, the Best Evidence Rule, also known as the Original Document Rule, applies. It can be summarized this way: "In proving the terms of a writing, where the terms are material, the original writing must be produced unless it is shown to be unavailable for some reason other than the serious fault of the proponent."11 The rule states that in order to prove the contents of a document, the original document, whenever possible, must be produced.12 Some states, such as Illinois, have modified this rule of evidence in certain circumstances to include reproductions if the original was destroyed in the "regular course of business."13

Computers operate by use of electronic signals and magnetic spots not visible to the human eye. This primary evidence cannot be observed by a jury or judge, the "triers of fact". Therefore, secondary evidence, in the form of computer-generated printed matter, will often be essential to

---

11David Bender, Computer Law Litigation, (New York, N.Y.: Matthew Bender 1985), Vol. 2, sec. 5.03(2).

12Spencer A. Gard, Jones On Evidence, (Rochester, N.Y.: The Lawyer Co-Operative Publishing Co. 1972), p.86, sec. 7:1.

13Ill. Rev. Stat. ch. 38 sec. 115-5 (1985).

successful prosecution.14

The first federal case to address the problem of using computer printouts as evidence was United States v. Russo.15 In Russo, the United States used the Federal Business Records Act, 28 U.S.C. sec. 1732(a), to introduce the printouts as business records. In affirming the introduction, the court stated, "Assuming that properly functioning computer equipment is used, once the reliability and trustworthiness has been established, the computer printouts should be received as evidence of the transactions covered by the input."16

In Illinois, the first case to address computer-generated evidence was People v. Gauer,17 which dealt with the admission of computer-generated phone records. In its ruling the court stated, "In light of the general use of electronic computing and recording equipment in the business world and the reliance of the business world on them, the scientific reliability of such machines can scarcely be questioned." Today Illinois, like many other states, has a statute18 that specifically deals with computer abuse.

14U. S. Department of Justice, Computer Crime: Criminal Justice Resource Manual, Bureau of Justice Statistics, Washington, DC: 1979, p. 114.

15480 F.2d 1228, (6th Cir. 1973), cert. denied, 414 U.S. 1157 (1974).

16United States v. Russo. 480 F.2d 1228, (6th Cir. 1973), cert. denied, 414 U.S. 1157 (1974), p. 1240.

177 Ill. App. 3d 512, 288 N.E.2d 24 (1972).

18Ill. Rev. Stat. ch. 38 sec. 16-9 (1985).

The courts have held that computer printouts can be used as evidence if a proper foundation is presented to ensure reliability and trustworthiness. Establishing a foundation requires a showing that:19

- o The computer records were made in the ordinary course of business.
- o The information on the records was placed in the computer within a reasonable time after the act or transaction to which it relates.
- o The information contained in the computer record comes from a source which itself is reliable.
- o The methods and circumstances of preparing the computer records provides reason for the trier of fact to believe that the information is reliable. 20

What Evidence Is Needed

The controlling criminal statute will dictate what evidence needs to be gathered. The prosecutor will decide which statutes apply and what data needs to be introduced as evidence at the trial. Special programs may have to be written to generate special audit reports. If special programs are written, the courts have held that the programs may have

---

19Jay J. Becker, "The Investigation of Computer Crime," Operation Guide To White-Collar Crime Enforcement, United States Department of Justice Report On White Collar Crime, April 1980, p. 30.

20Monarch Federal Savings and Loan v. Genser, 156 N.J. Super 107, 383 A.2d 475 (1977).

to be produced in court and their use justified.21

Initially, federal cases were handled under the approximately 40 potentially applicable federal criminal statutes, none of which mentioned "computers" or "software." State cases relied on similar statutes, drafted long before the advent of computers.

In United States v. Seidlitz,22 the government used the federal fraud-by-wire statute23 successfully because the defendant sent computer traffic across state lines by telephone. With new statutes that deal specifically with computers, prosecution is easier and evidentiary problems reduced. Many states have enacted laws directly addressing computer abuse. Nationally, President Reagan signed "The Counterfeit Access Device and Computer Fraud and Abuse Act of 1984" in October of 1984. This act amended Title 18 of the United States Code, and established specific penalties, including fines of up to $100,000 and imprisonment of up to 20 years, for specific types of computer abuse. The law also specifically defined a computer as "an electronic, magnetic, optical, electrochemical, or other high speed data processing device performing logical, arithmetic, or storage functions,

---

21United States v. Dioguardi, 428 F.2d 1033, (2nd. Cir. 1970), cert. denied, 400 U.S. 825 (1970).

22589 F.2d 152, (4th Cir. 1978), cert. denied, 441 U.S. 92 (1979).

2318 U.S.C. sec. 1343.

and includes any data storage facility or communications facility directly related to or operating in conjunction with such device."24

How Evidence Should Be Stored And Marked

For evidence to be admissible, the law requires that the proponent of the evidence demonstrate that the evidence is reliable and valid. This is referred to as establishing a foundation for the evidence. It is necessary to establish a foundation showing who has handled the evidence and where it has been stored. Great care must be exercised to maintain the "chain" of custody. The chain of custody rule requires the party seeking to introduce evidence to prove that the specimen or object was, in fact, derived or taken from a particular person or place.25 The rule also requires that an unbroken line of custody be demonstrated and proof presented that the integrity of the evidence was protected.

It may be necessary to show who made the backup tapes, who took the tapes to the vault, who had access to the vault, and so on. You must be able to show who has been in contact with the evidence from the time it was made until it was

---

24 18 U.S.C. sec. 1030.

25 Andre A. Moenssens, Ray Edward Moses and Fred E. Inbau, Scientific Evidence In Criminal Cases, (Mineola, N.Y.: The Foundation Press, Inc. 1973), pp. 18-19.

turned over to the police. To make sure the chain is preserved, the team's legal advisor must be involved in setting up these procedures.

As part of demonstrating the chain, the evidence is usually physically marked by each person taking possession of it. For example, if backup tapes or disks are made for an investigation, the operator or investigator should initial the tape, both on the reel and on the tape itself. The five feet of magnetic tape or leader tape can be handled or written on. The tapes should be then sealed, and the seal initialed. Backup disks should be marked on the bottom of the pack and then sealed in a container. Both should be stored in an area with a temperature range between 40F - 90F (4C - 32C) with the relative humidity between 20 percent and 80 percent.26 In this environment, tapes and disks can be stored up to three years.27

Punch cards and punch paper tape should be protected from rough handling and stored in sealed metal or cardboard boxes in the same environment as tapes and disks. They should be initialed and sealed in a container with the container's seal initialed. The storage life of cards under these conditions is indefinite.

26U. S. Department of Justice, Computer Crime: Criminal Justice Resource Manual, Bureau of Justice Statistics, Washington, DC: Fall, p. 111.

27Ibid., p. 111.

All evidence should be logged and kept in a secured area before it is turned over to the police. The area should have a log showing everyone who entered the area while the evidence was being held.

If the evidentiary chain is properly maintained, the evidentiary problems in court will be minimized.

III. Specific Techniques For The HP3000

There are several computer abuse prevention techniques that can be applied to the HP3000. The most vulnerable point or any computer are the dial-up lines. In the HP3000 environment, this can be addressed in many ways, For example, the ports can be logically "downed" via the DOWN command. If anyone wants to use the computer, they must check with the operations staff first.

Another method is to use hardware, a call-back device, smart cards with constantly changing hardware, encryption modems, or special keys.

Finally, there is software, generally in the form of a program that will check security that is activated via a logon UDC. This program can be developed in-house, or there are a couple of very good packages on the market.

One area where many installations are lax is in not changing passwords to support and third-party accounts, for example, TELESUP, ROBELLE, REGO and COGNOS. Many still have the same password as when the machine was installed.

Many installations do not control SM, PM, and OP capabilities. As Eugene Volokh pointed out in his paper

entitled, "Burn Before Reading - HP3000 Security And You," with PM you can get system manager capabilities in just a few key strokes!28

28Eugene Volokh, "Burn Before Reading - HP3000 Security And You," Thoughts and Discourses on HP3000 Software, (Los Angeles, CA: Vesoft), 1986, pp. 17-43.

IV. Summary

This paper has stressed the importance of pre-planning before a computer abuse is discovered. An installation should have a Computer Abuse Response Team in place long before problems occur. An integral part of the team should be the legal advisor. The legal advisor should make sure evidence is gathered correctly AND the installation is protected legally at all times.

Once formed, the Computer Abuse Response Team should meet with senior management to determine what the response to computer abuse will be. All members of the installation should know of the response team's existence. In this way, employees will know the installation takes computer abuse VERY seriously. The team's existence, in and of itself will have a deterring effect. The response team should institute procedures for obtaining a secured area for keeping evidence and maintaining the evidentiary chain should an abuse occur.

To keep abreast of the changing environment, several professional groups have been formed to educate managers and auditors in computer crime and abuse. One organization is the Computer Security Institute (43 Boston Post Road, Northborough, MA 01532). They publish the Computer Security

Journal, act as a clearinghouse on computer abuse and prevention, and provide training courses.

EDP Auditors Association, Inc., (373 Schmale Rd, Carol Stream, IL 60187) is an international organization providing certification and training for their members. They also publish The EDP Auditor Journal. Additionally, many public accounting firms now provide EDP auditing and consulting as part of their services.

In the past, the business community was reluctant to publicize the fact that computer abuse was taking place. Because of this reluctance, many cases were never brought to the attention of the police and the courts. Now, more and more businesses are prosecuting abuse cases, as are many universities, where some of the more "creative" students adjust grades or crash computers. The prosecution of this type of crime is sending a new message to computer hackers --- computer abuse is going to be taken very seriously.

As long as computer crime is profitable, it will continue. However, with proper pre-planning, improved training and an enhanced awareness of the problem, it can be combatted.

BIBLIOGRAPHY

American Bar Association
1984 "Report On Computer Crime", Task Force On Computer Crime, Section of Criminal Justice, June.

Bailey, F. Lee and Henry B. Rothblatt
1984 Defending Business and White Collar Crimes, Federal and State, Second Edition, Rochester, NY: The Lawyers Co-Operative Publishing Co.

Becker, Jay J.
1980 "The Investigation of Computer Crime", United States Department of Justice, report on White Collar Crime, April.

Bender, David
1985 Computer Law Litigation, New York, NY: Matthew Bender.

Carrol, John M.
1977 Computer Security, Los Angles, CA: Security World Publishing.

DeHetre, J. David
1976 "Computer-Generated Evidence --- Is It Different?" Chicago-Kent Law Review, Vol. 52, No. 3, pp. 567-599.

Enger, Norman L. and Paul W. Howerton
1980 Computer Security - A Management Audit Approach, New York, NY; AMACOM.

Gard, Spencer A.
1972 Jones On Evidence, Rochester, NY: The Lawyer Co-Operative Publishing Co.

Jenkins, Martha M.
1976 "Computer-Generated Evidence Specially Prepared For Use At Trail", Chicago-Kent Law Review, Vol. 52, No. 3, pp. 600-609.

Landreth, Bill
1985 Out of The Inner Circle, Bellevue, WA: Microsoft Press.

McKnight, Gerald
1974 Computer Crime, New York, NY: Walker Publishing Company, Inc.

Moenssens, Andre A., Ray Edward Moses and Fred E. Inbau
1973 Scientific Evidence In Criminal Cases, Mineola, NY: The Foundation Press, Inc.

O'Donoghue, Joseph
1987 "Strategies Found To Be Effective In The Control of Computer Crime In The Forbes 500 Corporations," SIG Security Audit & Review, New York, NY: ACM.

Parker, Donn B.
1983 Fighting Computer Crime, New York, NY: Charles Scribner & Sons

1976 Crime By Computer, New York, NY: Charles Scribner & Sons

Rochester, Jack B. and John Gantz
1983 The Naked Computer, New York, NY: William Morrow & Co.

Schweitzer, James A.
1982 Managing Information Security - A Program for the Electronic Information Age, Woburn, MA: Butterworth Publishers, Inc.

Sprowl, James A.
1976 "Evaluating the Credibility of Computer-Generated Evidence", Chicago-Kent Law Review, Vol. 52, No. 3, pp. 547-566.

U.S. Department of Justice
1982 Computer Crime: Electronic Fund Transfer Systems and Crime, Bureau of Justice Statistics, Washington, DC: July.

1982 Computer Crime: Computer Security Techniques, Bureau of Justice Statistics, Washington, DC.

1980 Computer Crime: Expert Witness Manual, Bureau of Justice Statistics, Washington, DC: Fall.

1980 Computer Crime: Legislative Resource Manual, Bureau of Justice Statistics, Washington, DC.

1979 Computer Crime: Criminal Justice Resource Manual, Bureau of Justice Statistics, Washington, DC.

1977 The Investigation of White-Collar Crime Law Enforcement Assistance Administration, Washington, DC: April.

Volokh, Eugene
1986 "Burn Before Reading - HP3000 Security and You,"

Thoughts and Discourse on HP3000 Software, Los Angles, CA: Vesoft, pp. 17-43.

Wong, Ken
1984 "Computer Crime in the UK - Are You Vulnerable?", Datenschutz und Datensicherung, January 1984, pp. 34-36.

1983 "Computer Crime - Risk Management and Computer Security", Datenschutz und Datensicherung, March 1983, pp. 256-259.

Younger, Irving
1975 "Computer Printouts in Evidence: Ten Objections and How to Overcome Them", Litigation, Vol. 2, No. 1, pp. 28-30.

Zajac, Bernard P., Jr.
1987 "Many Devices Can Protect Vulnerable Dial-up Lines", Government Computer News, February 13, 1987, p 47.

1986 "Une Criminalite Nouvelle Pour Une Technologie Nouvelle: Que Faire Lorsque Vous Avez Raison De Croire Que L'Integrete De Vorte Ordinateur A E'te Atteinte --- New Technology Means New Crime: What To Do If You Have Reason To Beleive Your Computer Has Been Compromised", Presentation to the 4th Worldwide Congress On Computer and Communications Security and Protection, Paris, France, 5 March 1986.

1985 "La Fraude Informatique Par Des E'tudiants: Analyse De Cas --- Computer Fraud in College - A Case Study", Presentation to the 3rd Worldwide Congress On Computer and Communications Security and Protection, Cannes, France, 6 March 1985. "Police Response to Computer Crime in the United States", The Computer Law and Security Report, July/Aug 1985.

"EFT Protection In The United States", The Computer Law and Security Report, Nov/Dec 1985.

Monarch Federal Savings and Loan v. Genser,
156 N.J.Super 107, 383 A.2d 475 (1977).

People v. Gauer
7 Ill. App. 3d 512, 288 N.E.2d 24 (1972).

United States v. Dioguardi, 428 F.2d 1033 (1970)
2nd Cir. 1970, cert. denied, 400 U.S. 825 (1970).

United States v. Russo, 480 F.2d 1228 (1973),
6th Cir. 1974, cert. denied, 414 U.S. 1157 (1974).

United States v. Seidlitz, 589 F.2d 152 (1978),
4th Cir. 1978, cert. denied, 441 U.S. 92 (1979).