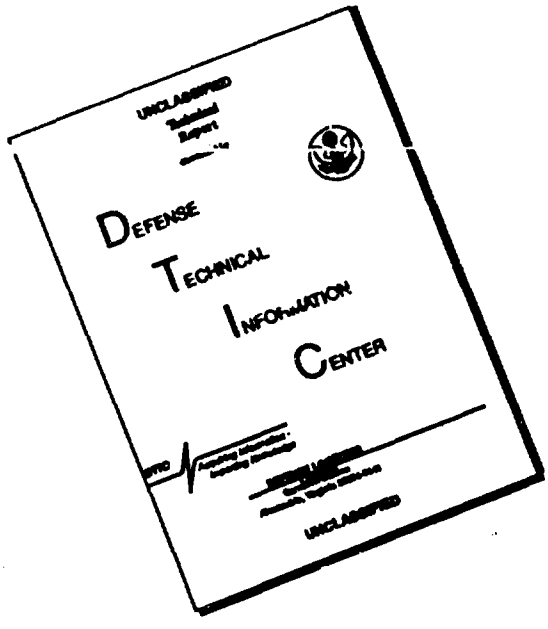
PROCEEDINGS OF THE SECOND ANNUAL WORLD-WIDE DATA MANAGEMENT SYSTEM SYMPOSIUM

SPONSORED BY THE UNITED STATES AIR FORCE ACADEMY

DEPARTMENT OF ASTRONAUTICS AND COMPUTER SCIENCE

DECEMBER 1971

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Department of Astronautics & Computer Science US Air Force Academy CO 80840	Unclassified
	2b. GROUP N/A

3. REPORT TITLE

Proceedings of the Second Annual Worldwide Data Management System Symposium

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Annual

5. AUTHOR(S) (First name, middle initial, last name)

Gordon M. Gerson, Major, USAF
Anthony J. Winkler, Captain, USAF (Editors)

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1971	139	47

8a. CONTRACT OR GRANT NO	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	N/A
c. N/A	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	N/A

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
N/A	Department of Astronautics and Computer Science US Air Force Academy CO 80840

13. ABSTRACT

These proceedings provide specific information regarding four U S Air Force Data Management Systems: Personnel, AFOLDS, ADVISOR, GIM. In addition, there are three papers concerned with system architecture, system test methodology, and system selection and evaluation. All of these papers were presented at the Second Annual Worldwide Data Management System Symposium, held at the United States Air Force Academy, Colorado on 9, 10, and 11 December 1971. (U)

DD FORM 1 NOV 68 1473

Unclassified

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Data Management System						
U S Air Force						
System Architecture						
Data Base Management System						
Personnel System						
Evaluation						
On Line Data System						
System Design						
Computer						
File Management						

Preface

The Second Annual Worldwide Data Management System Symposium was held at the United States Air Force Academy on 9, 10 and 11 December 1971 under the sponsorship of the Department of Astronautics and Computer Science of the United States Air Force Academy.

The first symposium, held in November 1970, was composed of briefings about various military data base management systems, and did not involve the formal presentation of papers concerning current areas of study. Thus, no proceedings were published.

This publication represents what we hope will be the first of a series of informative documents regarding the current activities of USAF organizations in the field of Data Base Management Systems.

Due to scheduling conflicts, it has been decided to hold future symposia in the early part of each calendar year. The third such annual symposium has been tentatively scheduled for early 1973.

The editors wish to thank all of the participants for their contribution toward the success of this meeting.

Gordon M. Gerson, Major, USAF
Anthony J. Winkler, Captain, USAF

United States Air Force Academy

CONTENTS

The MITRE Corporation
Bedford, Massachusetts

CURRENT TRENDS IN DATA MANAGEMENT SYSTEM ARCHITECTURE

Jonathan A. Singer
Command and Management Systems

Sponsorship - Air Force Contract Number F19628-71-C-0002

INTRODUCTION

The views expressed in this paper are derived to a great extent from the author's involvement in both past and present data management projects sponsored by the Electronic Systems Division of the Air Force. These projects span a period of more than ten years during which there has been considerable change in the field of data management. The most significant single change is the growing acceptance of data management as an essential set of tools for the management of formatted files. Ten years ago, virtually all significant data management development work was being done within the Department of Defense community, largely in response to the demanding requirements of command and control systems. The management support users of data processing, both in the military and commercial environments, regarded data management systems as being inappropriate tools for their immediate needs. The reasons for this are fairly straightforward. First, the demands for high speed data retrieval and multi-user access were critical needs in the command and control environment and strongly encouraged the development of such systems. Second, the cost of such systems was very high. They typically required over a million dollars to implement, and took a number of years to complete. This level of investment was out of range with that which the management support community could afford. Additional impetus for the building of these large, high-performance systems was provided through the availability in the command and control environment of what was then large sophisticated hardware such as the SAGE computers. Many non-command and control users of data processing did show an interest in data management, however, their available resources limited this interest to using fairly simple tape file extraction and report generation programs.

This early situation is in strong contrast with that of the present. There are over one hundred data management systems being marketed today largely as proprietary software, and there is a dramatic increase in the use of these systems by commercial users of data processing. The Informatics Inc. MARK IV system, for example, has a client list of over 400, while IBM's IMS system is over 200. Within the set of DOD systems, NIPS has found wide acceptance in terms of the number of installations where it is being used.

The most significant conclusion to be drawn from these numbers is that data management is no longer regarded as the exclusive property of command and control, but as a necessary and even vital part of other segments of the data processing user community. More specifically, the management support segment which is dominant in both the military and commercial environments has begun to greatly expand its interest in and use of sophisticated data management capabilities.

This wide use, more than any other single factor, has begun to influence not only the range of jobs to which data management is being applied, but the architecture of the systems themselves. This is not surprising. In any field, increasing demand for a product or service has a strong influence on the basic economics of providing that product or service, and a subsequent change in the methods of constructing and supplying it. Probably the most obvious example of this in data processing is the history of operating systems. Early operating systems were little more than simple utilities which functioned as aids in using the hardware. Today they are a necessity. Ten years ago it was extremely difficult to imagine a manufacturer supplying as a standard operating system something as extensive and complex as the System/360 Operating System. Now the capabilities of such systems are regarded as commonplace, and users have developed even higher expectations for the future.

The motivation for the development of these vastly improved operating systems has rested largely on the growth of the data processing industry. This has allowed manufacturers to make major investments to develop such complex systems and to amortize their cost over a much wider market and a long period of time.

Data management systems have now begun a transition very similar to that of operating systems. They are still largely in the category of a utility, albeit a rather sophisticated one. The volume of demand for their capabilities, however, is sufficiently high so that new DMS architectures for meeting this demand efficiently can now be realistically considered.

The central thesis of this paper is that the most effective way of supplying data management capabilities in the future is to imbed a substantial number of the components of a DMS within a modern operating system. The specific components chosen in this paper are those which, in addition to being common to every DMS, are sufficiently alike in current data management systems to warrant use of a single implementation of that component in future systems. It is argued that this approach provides the following benefits:

i. Any component of a data management system can be made to operate more efficiently and effectively when it is included as part of the operating system.

ii. Many DMS components and the functions they support are not unique to data management systems but are common to many other types of processing. Consequently, inclusion of DMS components within the operating system makes them more widely available, and promotes compatibility between the data management system and other non-DMS programs.

iii. Given adequate implementation of these components, the designers of data management systems will be free to concentrate on providing better capabilities in support of specific DMS applications.

A straightforward way of defending these statements is to describe the problems faced by designers of early data management systems in terms of the tools available to them, and to compare these problems to those faced by current designers and implementors of data management systems. This comparison reveals a trend toward providing many facilities which are essential in the implementation of data management systems within current operating systems. By extrapolation of this trend, a probable architecture for future data management systems can be described. This description substantiates the main thesis of this paper.

DMS IN THE PAST

The most striking problem facing designers and implementors of data management systems in the early 1960's was the lack of almost any basic support software useful for building a data management system.

The author of this paper was a member of a group which implemented a data management system on the SAGE computer, the AN-FSQ-7, in 1961. This machine had extremely impressive hardware capabilities for its time, and provided an excellent vehicle for constructing a data management system. It had 65k, 32 bit words of 6μs core storage, high speed drum storage with an access time of under twenty milliseconds, a communications drum which allowed for simultaneous use up to 10 teletypes, graphics display consoles, and high speed electrostatic printers. This system, however, was devoid of any general purpose software. There was no operating system or utility package available, other than a symbolic assembler and loader.

The data management system which was built, called ETF, was an experimental system and was not intended for operational use, although it was used as a base for prototype applications in military airlift operations and in post-attack command and control. The data management capabilities which this system provided included a general purpose query language and output formatting routines, file structures providing rapid retrieval through a hash-code technique and multi-terminal on-line operation. In many respects, this system was comparable in capabilities and performance to some that are being marketed today. The process of implementing it, however, was considerably different.

This implementation process can be conveniently divided into three phases. First was the implementation of very low-level routines such as input-output handling and core storage allocation. These can be thought of as "basic operating system components". Today they are taken completely for granted. Second were file and dictionary access programs, file storage allocators for management of disk space, and routines to manage multiple simultaneous teletype input-output including management of task queues. These routines can be termed "basic DMS components". At the time, most of these capabilities were quite new and were largely untried. Today, they are taken for granted in almost every DMS, although they are built using a variety of implementation techniques.

Once the capabilities described above had been provided, the third phase could begin. This phase concentrated on the implementation of the query language translator, output formatting routines, special computational routines and the file generation program. These programs can be termed "basic language components". This division of a DMS into components can be further clarified by thinking of them in terms of those which are transparent to a user of the DMS and those which are visible to him. The operating system components and basic DMS components are transparent to a user of the system, although they have a strong influence on its performance. The basic language components constitute the user's interface with the data management system and strongly influence the suitability of the system for various applications.

A further observation about the ETF system is that the basic language components of the system were by comparison easier to construct than were the basic DMS components. This is true because the presence of an adequate set of basic components allows for substantial change within the language components of the system even after their initial implementation. The most important observation, however, is that virtually the entire system had to be built from scratch.

CURRENT DATA MANAGEMENT SITUATION

All of the goals which the designers of the ETF system were trying to achieve are still very valid. Such capabilities as on-line query languages supported by high-speed retrieval mechanisms are characteristics which many systems today either contain or aspire to. The MRI Inc. System 2000, Cambridge Computer Associates CCA104 System and TRW's GIM System are all contemporary systems which were designed with these characteristics as major goals. Systems such as IBM's NIPS and Informatics MARK IV which were fairly simple batch processing

systems in their original implementations have evolved upwards in complexity to include many of the features of newer higher performance systems.

In addition to similar goals, the design techniques used in current systems are remarkably like those employed in early systems. This is not to say that there have not been improvements, because there have, however, such file access methods as hash-coding, inversion of files using bit vector schemes or pointers had all been implemented at least once by 1962. Current query languages also show remarkable similarity to those of early systems. For instance, the language of SDC's DS/2 system bears more than a vague resemblance to that of the ETF system.

Where have the major advances in data management systems ocurred, then? There have been no dramatic breakthroughs in the field of technical design. Improvements have been evolutionary in nature and have come largely through reimplementation and refinement of basic techniques. The most significant advance in data management systems is their acceptance by a much broader segment of the data processing industry than was formerly the case, and the dramatic rise in the number of available systems. Much of the reason for this availability is due to the inclusion of basic tools within the operating system, similar to the basic DMS components described above, which makes implementation of data management systems quicker and cheaper.

There are a number of examples of basic DMS components which are contained in current operating systems. The most important of these is physical file access methods. It is important because the designer of any data management system must decide whether to make use of access methods which have been provided with the operating system or to design and implement his own. The cost differential between these alternatives is large, as is the implementation time. Because of this, many data management systems make use of these access methods Within the IBM 360/370 environment, the NIPS system utilizes the Indexed Sequential Access Method for organizing its data files. This is also true for the Systems Development Corporation's DS/2 system which offers the option of using either sequential or indexed sequential access methods. The INQUIRE data management system built by Infodata Systems, Inc., for use on IBM 360/370 equipment, uses a combination of indexed sequential and direct access methods. The data management system bid by Honeywell Information Systems in its WWMCCS submission provides an additional example. Originally, this system was designed to handle tape files compatible with a number of other systems including COBOL. As a part of its enhancement for the WWMMCS bid, interfaces between the system and additional access methods available under GECOS III were built. These include both

the random file capability as well as the recently announced Honeywell Indexed Sequential Access Method.

There are numerous other examples of the type cited above, all of which serve to demonstrate that current data management system designers frequently make extensive use of manufacturer supplied access methods.

A second significant area in which operating system capabilities are used to support data management functions is in teleprocessing. The use of IBM's teleprocessing access methods by many existing data management systems is an example of this. While these capabilities are rather basic, the availability of the Time Sharing Option of OS 360 provides a much more substantial set of such tools and will certainly be used in future systems. This can be illustrated most clearly with an example drawn from the author's recent experience.

The Air Force Data Services Center (AF/ACS) established a project in 1970 to acquire an interim data management system. The system which was chosen had been built by General Electric Apollo Systems Department for NASA and was called ADVISOR. The system had been implemented originally under GECOS II, which had no time-sharing subsystem. The system, however, was intended to support multiple on-line terminals. The implementors were naturally forced to develop a sub-monitor of their own since GECOS II provided no support for subsystems with terminal capabilities. The GECOS II version of ADVISOR, including its sub-monitor, required 34K words of core storage. Furthermore, it was necessary to alter the job scheduling algorithms of the GECOS II operating system to guarantee adequate response times to ADVISOR terminals. At the time the ADVISOR system was selected for installation at AF/ACS, GECOS III with its time-sharing subsystem had become available, and it was decided to modify the ADVISOR system to run under time-sharing.

The modifications to the system resulted in substantial improvement in two areas. First, the amount of core required by the system went from 34K words to 23K words, a reduction of almost one-third. Second, the response time at terminals was significantly better in the modified version even when the DMS was competing with a number of other time-sharing users.

It is the author's belief that this experience is representative of the kinds of savings which could be realized in many data management systems by using time-sharing facilities supplied as a part of the operating system. This does not mean that current time-sharing systems provide all features needed for support of data management systems. However, they represent a significant advance over tools which have been available in the past.

CURRENT TREND

The examples presented above have illustrated the trend towards including many basic DMS components necessary to data management within operating systems. Clearly, some of these components would exist in the absence of any DMS requirements. However, a large enough number of them have been provided specifically for data management related functions so that a precedent has been established. This raises two additional questions. First, will this trend continue? And, second, should it continue?

Will Trend Continue?

The answer to the first question rests on two main issues: the additional needs of data management system designers in terms of basic DMS components required, and the size of the market perceived for these by suppliers of operating systems. The usefulness of additional components for use in constructing data management systems can be shown by an examination of a number of specific representative areas in which such components are now being developed. These are discussed in more detail in the next section. The accurate determination of the size of future markets for these components is more difficult. The president of IBM, Mr. Cary, in a recent article[1] is quoted as saying that systems incorporating a data base and data communications have the most future market potential. A major manufacturer of data processing equipment is known to be developing a large array of tools for data base manipulation which are architecturally within the operating system. At a general level, the increase in the amount of formatted file processing and more specifically the processing of large shared data bases, is an established fact. The wide use of systems such as IBM's IMS and the CINCOM's TOTAL system as a set of basic DMS components for data base management systems bears this out.

Should Trend Continue?

The second question raised above; namely, the desirability of a continuation of this trend is largely a subjective question. In discussing data management system design with collegues, this author has more than once heard the argument advanced that "operating systems are making it difficult to implement many of the low-level detailed functions required by data management". This is undoubtedly true, and is in fact additional evidence that the trend cited in this paper is established. These people, to a great extent, are concerned with the specific problem of having to deal with standard access methods which do tend, in some instances, to make it difficult to develop complex new file structures. The alternative to this is to

provide features within the operating system which will allow implementors to start from scratch at a "bare bones" level and build their own systems. This is incompatible with the fundamental philosophy of modern operating systems. Multiprogramming systems, for example, require that users give up some individual freedoms (e.g., first-level interrupt handling) in the interests of overall system efficiency. In the case of file access methods, their use on a wide basis indicates the existence of a concensus that the availability of a standard set of them is sufficiently valuable to warrant a compromise in flexibility. None of these statements is intended to suggest that data management system designers should be prevented from experimenting with new techniques. This, in fact, should be encouraged. It does mean, however, that much of this experimentation may have to be carried out external to the environment of standard operating systems. In short, the trend toward the wider availability of basic DMS components provides useful tools to implementors of operational data management systems on standard hardware and software and can sometimes be a hinderance to those designers whose interests lies solely in experimentation with new techniques on these same systems.

To return to the main question at hand, the established trend toward placing basic DMS components in operating systems should continue because it shows every indication of further lowering the implementation cost of data management systems, allowing the implementors of the DMS to concentrate on basic language components, and increasing the compatibility among all software elements of any facility using the standard operating system. The following section describes a number of specific technical areas on which a continuation of the trend is likely to be based.

SOFTWARE CAPABILITIES FOR INCLUSION IN FUTURE OPERATING SYSTEMS

Each of the topics discussed below is an area in which there is currently one or more implementation activities underway or which are being widely and actively studied. These areas can be divided into two main types: those which were originally motivated by or developed specifically for data management use, and those which were or are being developed for general use but are critically important to future data management systems. In either case, they represent basic DMS components which belong within an operating system.

Improved File Structures

The need for increasingly sophisticated file structures is well established. The inclusion of support for these structures within

the operating system is advantageous for two main reasons. First, it makes such structures available to all processing functions, including procedure oriented languages. Second, such structures can be provided through extension of current operating system capabilities. These extensions are described in detail below.

Before discussing specific file structures, definitions of logical and physical file structures are appropriate, since there is often confusion over the distinction between these terms. For the purposes of this paper, a logical file structure is the set of expressed or implied relationships between records or entities within a file or data base. Thus, such terms as "flat files", "hierarchical files", and "network structured files" all describe logical file structures. Physical file structures are the mechanisms utilized to store and retrieve file data. These include such techniques as sequential access methods, direct access methods, bit vector schemes for file inversion, and others. It should be noted that there is usually, but not necessarily, a one to one correspondence between physical and logical file structures. For example, an indexed sequential access method might be used to store either flat or hierarchical files.

Symbolic Reference to Data

A conceptually simple extension of current operating systems' file access methods is that of symbolic field-level reference to data. All current access methods are organized at the lowest level around the notion of records. In only limited cases (the index of ISAM is the most obvious example) is the access method sensitive to units of information at a level lower than that of a record. Both data management systems and formatted file related applications programs operate primarily on field level information which in most cases is represented as contiguous strings of characters within a record. Furthermore, most non-DMS applications programs are bound to specific record formats in the sense that any change in either field level information or the physical file structure causes the program to run incorrectly. In the past, this situation was undesirable but could be tolerated, since many application programs were the sole users of a data file. In a shared data environment, this method of binding programs to specific formats is highly undesirable. Data formats are changed quite frequently, forcing application programs to change with them. What is needed is an ability to maintain a single description of a record's contents rather than allowing each applications programmer to maintain a separate and unique data declaration. This situation has been available in COBOL by using external data divisions, however, it is rarely taken advantage of.

A more desirable mechanism for both data management systems and non-DMS applications programs is a physical data description or data declaration managed by the operating system as a natural extension of current access methods. This is very similar in concept to the dictionary of current data management systems, and its inclusion as a part of file manipulation routines makes its advantages more widely available. An applications programmer or data management system making use of this capability would invoke the name or identifier of the record type it desired access to, and after reading any record of this type, the fields of that record could be referenced symbolically. The most obvious way of providing this capability in the future is to extend, to the field level, the file cataloging and file directory services provided by current third-generation operating systems.

Physical File Structures

The key to the performance of any data management system is the physical file structure it employs. Because of this, designers of data management systems have devoted more time and thought to the relationship between logical and physical file structures than any other single component in a DMS. There are a number of generally accepted maxims which have evolved from the observation of many implementations of different types of file structures. The most important by far is that there is no single access technique or physical file structure which provides optimum performance over a wide range of common file processing applications. For example, demands for very high speed retrieval from large files are incompatible with those of large volume, rapid updating, and high volume output requests. To a great extent, this is a reflection of the limitations of current secondary storage devices, and it points out the needs for a diversity of access methods from which the most suitable one for any application can be selected.

Because no single physical access method stands out as being universally better than all others, it is difficult to propose a specific one, or even a small set for inclusion in future operating systems. It is clear, however, that demand for increasingly complex physical structures is increasing, indexed sequential and direct access being the best examples, and is likely to continue. Perhaps the most obvious next steps in complexity are those of multiple indexes for a single file (file inversion) and the provision of physical structures which support logical structures of increased complexity.

File inversion techniques, and there are many of them, are the most frequently used mechanism for providing high-speed retrieval within the limitations of currently available hardware. For applications requiring this type of performance, a rapid method of selecting records based on their content is necessary. To either a data

management system designer or to an applications programmer, the capability would take the general form of a call to the operating system which delivers to the user the "next" record or records whose contents meet the criteria specified in the call. The specific physical access methods used within the operating system to achieve this should be largely transparent to the program making the call. Furthermore, it should be mentioned that the usefulness of such a capability is made possible by the availability of the file dictionary described previously, since any qualifying field can be referenced symbolically.

Logical File Structures

In addition to improved physical access methods, an ability to manipulate file structures of greater logical complexity is needed. This need is currently being met by such systems as IMS and TOTAL, which support hierarchical and network structured files. An example of the need for these structures can be found in the Military Airlift Command's MACIMS system. One of the primary goals of this system is the management of a large body of interrelated data. This includes a reduction in the redundancy of data storage, and the ability to provide data to a number of different functional processors, each requiring a different but, not necessarily unique, subset of the data. Comparison of these requirements with the abilities of traditional file structures such as that supported by COBOL reveals a significant gap between requirements and capabilities, and thus a need for structures such as those provided by IMS.

The implementation of support for such structures within an operating system is reasonably straightforward technically, and can be constructed using the dictionary and some of the physical access capabilities proposed earlier. The most difficult problem currently being faced is that of arriving at a consensus as to which set of logical and physical structures are best. It seems likely that much of the discussion is due to a lack of sufficient data on the relative merits of one structure versus another, and that any firm decision cannot be made until such data is generated.

Regardless of what structures are chosen, the effects on the operating system and the types of support it must provide are much the same. At a detailed level, the capability needed is that of retrieving a record from a file based on a logical relationship of of that record to another in the file. Thus, a simple example is one of moving downward in a hierarchical tree to retrieve the offspring records of a parent record. Again an important point about placing these capabilities within the operating system is that they become available not only to data management system designers and implementors

as a set of capabilities which form a base on which a data management system can be constructed, but also to the application programmer as a basic extension of his programming language.

Additional Basic DMS Components

The discussion above on file structures is largely centered on techniques which have been developed within the domain of data management, but which today find much broader applicability. The additional areas discussed below originated in a wider context than DMS but are critical to their operation and which most sensibly (from a technical and economic point of view) belong in the domain of the modern operating system.

Concurrent Task Management

Many data management systems are designed with at least some thought of providing service to multiple simultaneous on-line users. As the examples in the earlier portions of this paper have shown, one promising vehicle for providing this capability is the time-sharing subsystem of many modern operating systems.

Unfortunately, current time-sharing systems are not entirely adequate for supporting all of the concurrent task management needs of a data management system. In particular, most time-sharing systems have been developed to support independent, unrelated jobs. This is incompatible with many needs of current data management systems. A prime example is the area of related job scheduling. In many data management applications, it is desirable to be able to assign priority to different types of users or the jobs which they are performing. This need is usually met by providing a method of queueing a set of jobs all operating on a single file, and interrupting tasks which are in execution to service, on arrival, higher priority tasks. Current time-sharing systems must be improved substantially in this area to meet the needs of data management systems. An additional improvement to time-sharing systems which would benefit not only data management systems but all users of time-sharing is that of reentrant program support. In a data management system environment, there is a high probability that more than one user is executing the same module of the data management system. For example, in a system with ten simultaneous on-line users, it is very likely that more than one of them is making use of the query language translator. This is a situation where reentrant programs can provide a significant saving in core space required. In the operation of some current data management systems on IBM 360 equipment, it is always distressing to be forced to maintain a separate copy of the data management system in core for each system user. The effective use of reentrant programs running in a time-sharing environment can eliminate much of this problem.

Current time-sharing systems, then, must be improved to allow data management systems to operate at a high level of efficiency, and with a full set of capabilities. Nevertheless, the use of current time-sharing systems offers considerable advantage over the alternative of building similar capabilities nearly from scratch.

Management of Concurrent File Access

The problem of managing concurrent file access, including both reading and writing of files, has no general solution. This is a problem which is of interest to not only data management systems, but any set of programs referencing a common set of data files. There are a number of examples which can be used to illustrate the problem, probably the simplest of which is the case where program A is updating a file, and program B is reading it. Both programs are operating in a multiprogramming environment where either may be interrupted by the operating system without notice. Program A, then, may be stopped in the midst of an update which changes not only a data value, but the structural integrity of the file. Program B, in trying to read the file, may find a logically inconsistent data structure which it cannot sensibly process.

The general solution to this problem at the level of small units of file access, such as records, is not known. However, it is clear that any solution will be an integral part of the operating system. It must be tightly bound to the file access mechanisms of the system in order to maintain an awareness of all accesses to a given file. One can imagine a central, perhaps reentrant, routine which maintains control tables for all file access by any program or system. It is in a position to detect conflicts, and schedule file accesses in such a way that the conflict is eliminated. The need for such a solution becomes more critical as shared data applications come into wider use. Current solutions to the problem such as preventing all file access from being made while an update process runs to completion simply are not satisfactory.

Secure Data Management System Operation

A problem which is of obvious special interest to the DOD community is that of providing secure operation in a multi-access computing facility. This problem is a general one, and affects not only data management systems, but all programs running in a facility handling classified information. The problem is of particular interest to designers, implementors, and users of data management systems, since the central objective of a secure facility is protection of file data from unauthorized or accidental disclosure. Like all other functions discussed in this section, a majority of the mechanisms for providing

secure operation must be within the operating system. The reason for this is straightforward. Assume the existence of a data management system with an arbitrary number of security controls built into it. Also assume that this system runs under the control of an operating system which has not been designed to be secure. In such a situation, it is a simple matter for any person with a system programmer's knowledge to access data belonging to the data management system through independent means. Consequently, a secure data management system is one which maintains its own security controls, but depends primarily on its operating system for secure operation.

EFFECTS OF PLACING BASIC DATA MANAGEMENT SYSTEM COMPONENTS WITHIN THE OPERATING SYSTEM

The value to data management system designers and implementors of the capabilities discussed above can be shown by considering again the process of constructing the capabilities provided in the early ETF system, and comparing this with the original example. The most dramatic change is the virtual absence of both the first and second phases of the original implementation. The first phase is made unnecessary because all of the functions originally implemented in it are available today in modern operating systems. The second phase has been largely eliminated because the basic DMS components would be available within the operating system. This phase now becomes largely a process of selecting from available components, those most appropriate to the characteristics of the system being constructed. In the case of ETF, the first step would be the selection of a prime access method. Any key value or index method providing rapid access to individual records by name would likely duplicate (or better) the performance of the original system. The logical structure required would be a two-level hierarchical file, which compared to some structures supported even today, such as in IMS, is nearly trivial, and would certainly be available within the standard structures supported in the future. Having chosen one (or possibly more than one) access method and determined that the file structures desired were supported, design could begin on the query language translator, output formatting routines, retrieval processors, and file generation routines. The development of these programs could be done using the same time-sharing system that would ultimately support the data management routines as one of its subsystems. This is particularly advantageous since time-sharing is an excellent tool for program development and can reduce the elapsed time involved by 50 per cent or more.[2,3]

After completing these functions and installing those that interfaced with on-line users under the time-sharing system, the new ETF system would be complete. In the author's opinion, this procedure would provide a savings in time and manpower of roughly one-half over that of the original system.

In addition to the savings in implementation costs, the system of the previous example and for that matter any data management system built from the same basic capabilities would have a number of significant additional advantages over current and previous systems. First is the relationship between the data management system and standard programming languages. Since both the DMS and standard compilers would use the dictionary capability described earlier, data files operated on by the data management system could also be directly accessed from any programming language within the system. In the past, this sort of compatibility at the data level has often been desirable but largely unavailable. No problem-oriented language such as those provided in many present data management systems is capable of supporting very complex processing. Occasionally, this type of processing is necessary. However, in any data management system with a unique file structure, processing of its data with external procedure oriented language programs becomes nearly impossible. The common use of the data dictionary eliminates much of this problem.

The second advantage of data management systems built with basic capabilities within the operating system, is that of better overall stability. Any capability included as part of a standard operating system receives better maintenance, is better documented, and has better training associated with it than it would as a user generated collection of software. This is largely due to economy of scale; widely used software generates increased revenues, however, maintenance and documentation costs are largely independent of the number of users of the software.

CONCLUSIONS

Data management today is experiencing more rapid acceptance of its capabilities than has ever been the case in the past. The experience gained within the DOD community over the past ten years in DMS has been responsible to a great extent for this situation. This acceptance provides the possibility of altering the basic architecture of data management systems so that many of its components can be included within the operating system, thereby providing better support and wider availability for them. The inclusion of many basic DMS capabilities in the operating system can only be done successfully, however, by selecting broadly useful techniques which have been proven out in a number of implementations in the past.

The overall goal of the DMS community should be to make data management a naturally available tool within any data processing facility rather than a separate, special purpose collection of software as it has so often been in the past. Its potential value to a broad range of data processing problems is now established and accepted. The realization of this value is the most important problem facing data management.

REFERENCES

1. A. Pantages, R. B. Frost, "IBM: Changes at the Top" Datamation Vol. 17, No. 21, November 1, 1971, pp. 26-28.

2. M. Schatzoff, R. Tsao, R. Wiig, "An Experimental Comparison of Time-Sharing and Batch Processing", Communications of the ACM, May 1967.

3. H. Sackman, W. J. Erikson, E. E. Grant, "Exploratory Experimental Studies Comparing Online and Offline Programming Performance", Communications of the ACM, January 1968.

BIBLIOGRAPHY

System Development Corporation, DS/2 Users Manual, Santa Monica, California, 1970.

Computer Corporation of America, Technical Reference Manual, Series 100, Information Retrieval Software Systems, Models: 102, 104, Cambridge, Massachusetts, October 1, 1970.

TRW Systems Group, GIM System Summary, TRW Document No. 3181-A, Revision 01, Redondo Beach, California, 15 August 1969.

IBM Federal Systems Division, System/360 Formatted File System (NIPS), Vol. I-VIII, 30 September 1969.

National Aeronautics and Space Administration, MA001-013-1, ADVISOR MSF-DPS Familiarization Manual, Washington, D. C., January 1970.

IBM Corporation, GH20-0765-1, Information Management System/360, Version 2 General Information Manual, 1971.

Cincom Systems, Incorporated, TOTAL - the Data Base Management System, Reference Manual, Edition II, Version I, 1971.

Informatics Inc., Document No. SP-70-810-200, MARK IV File Management System, 1970.

MRI Systems Corpration, Document No. RM S2K 2.1, System 2000 Reference Manual, 15 July 1971.

Infodata Systems Inc., Inquire Technical Summary, April 1970.

IBM Corporation, Document No. GC26-3746-0, IBM System/360 Operating System Data Management Services, January 1971.

Connolly, J. T., Operational Considerations in the Use of Data Management Systems, The MITRE Corporation, Bedford, Massachusetts, (to be published)

CODASYL Systems Committee, Feature Analysis of Generalized Data Base Management Systems, Association for Computing Machinery, New York, New York, April 1971.

CODASYL, Data Base Task Group Report to the CODASYL Programming Language Committee, Association for Computing Machinery, New York, New York, April 1971.

USAF Personnel Data Base Management System

Thomas H. Lee, Major, USAF
Harry M. Kepner, Major, USAF
USAF Military Personnel Center
Randolph Air Force Base, Texas

General data management systems are seldom sufficiently general to be used without a large amount of tailoring. The design and evaluation concepts being used in the Air Force Advanced Personnel Data System Data Base Management System are evolving from consideration of available features from existing systems, vendor capabilities and unique development efforts.

The IM-1 data management system was considered, specification for vendor capabilities detailed, and evaluation of the current personnel system was made.

The tools of benchmarking, performance analysis, and simulation are recommended for evaluation. A necessary, but currently not available, design tool is the performance chart. The need to relate data use with hardware and software processes in data management is pointed out as an important performance consideration.

KEY WORDS AND PHRASES: system design, data management, data base management, personnel data system, file binding, file structures, system evaluation, data use accounting, system simulation, performance evaluation, data administrator.

INTRODUCTION

This paper presents some of the considerations, decisions, and plans which have been developed for the data management system for the Air Force Advanced Personnel Data System (APDS). A DMS design group was formed in July 1971 to establish design objectives commensurate with the requirements of APDS and to evaluate the capabilities of existing and proposed systems. A review of the current personnel data systems and the proposed system is necessary to gain a full appreciation of the APDS data management problems. Our interpretation

of the data management function and the design factors in the APDS environment will form the bulk of this paper. The final section consists of some notes on DMS evaluation including some suggestions for performance measurement studies. It is premature to make any specific conclusions with respect to the final configuration of the APDS Data Management System. Hopefully, a consistent design philosophy will prevail throughout the paper. We are not committed to any particular concepts, nor have we completed our examination of existing systems. There is no intent to provide any specific answers. In fact, we believe meaningful research at this stage of DMS evolution may produce more questions than answers.

Personnel Data Systems

A significant aspect of the USAF Personnel Data System is that it has been a system of growth and evolution. It has developed through the years in response to the needs of personnel management as these needs were identified. The "vertical structure" (the flow of data thru levels of command) of the data system grew as the need for broader control of personnel became apparent. The personnel management function has evolved from the squadron commander to the Consolidated Base Personnel Offices and eventually to the Military Personnel Center (MPC). The development of automated data systems through the years has been one of successively strengthening the weakest link; hence, characteristics of many eras of data processing still exist. The newest system, the base level military personnel system (BLMPS), is a terminal and disk-oriented system. Since the base is nearest the data source, this system is concerned with validating and editing data inputs. The files maintained at this level consist of 3500 characters for each officer and airman. Transactions in the form of card image records are prepared for transmissions to Major Air Commands (MAJCOM's) and the Military Personnel Center via Automatic Digital Network (AUTODIN). At the MAJCOM level, processing is done in a magnetic tape environment. The files maintained there are records of 1600 characters for each officer and 720 for each airman. In today's system the data flow is primarily from base to MAJCOM to the MPC.

At the MPC the systems for officers and airmen have been developed alternately beginning with PDS-O-65. These systems are basically magnetic tape/serial processing systems. Current data bases are 130,000 records of 1688 characters for officers and 625,000 records of 648 characters for airmen. PDS-O and PDS-A run independently on separate B5500 computers. The center also operates a third computer (H1250) for the Weighted Airman Promotion System. All three will be replaced by the APDS computer. As an interim measure the officer system is undergoing a design change, Project 2.5, which will become operational this year. This system derived its name from the "third generation data base management and retrieval schemes" and the "second generation computer." It is designed as a disk/random processing system but within the severe constraints of current hardware.

The Advanced Personnel Data System (APDS) [4] has been designed to fulfill the following general personnel management objectives:

1. Total Force Management: The capability to more effectively manage and administer the total available personnel resource (active duty USAF, Air Reserve Forces and USAF civilian employees).

2. Centralized Personnel Management: The establishment of a central repository and processing center for personnel data at Headquarters level in order to reduce the need for redundant data storage and processing at base and major command levels.

3. Predictive Management: The capability to predict, plan and manage future events through the employment of advanced modeling and simulation techniques with accurate and timely data.

4. Personalized Management: The capability to provide individualized career development and guidance to all active duty military personnel.

Under APDS, most data will flow directly to the Center from the bases with feedback data supplied to the MAJCOM's. The record sizes change at all levels. The bases will have 2388 character records for officers and 2249 for airmen, MAJCOM's 200 for officers - 200 for airmen, and the MPC 4687 for officers - 2985 for airmen.

These numbers indicate the size of the data base which must be managed. The form of the APDS data base management system is being influenced by current systems design and new concepts in management of large dynamic data bases. The final DMS will be specifically suited for a personnel data system.

From time to time, the question arises as to the difference between a personnel data system and any other inventory system. These attributes are the most significant in differentiating: first, the units (people) represented in a personnel system change characteristics often; secondly, no two are exactly alike, and finally, the concepts and techniques of managing people are never static. You don't worry about dependent travel when you transfer a propeller. The form and content of personnel management information changes to meet current needs. These changing attributes demand system flexibility.

Data Management System Design

The initial problem was to establish the boundary of the "DMS." Specifically, what tasks are included in a data management system and how are the responsibilities assigned. Some initial semantic stumbling blocks can be removed.

The terminology "data management system" is most often used in relation to integrated data bases. It implies services directed toward the organization and manipulation of data that is used throughout a system. Effectively, DMS is a set of software processes that provide more flexibility in systems operation and more efficiency in the performance of generalized activities. For Advanced Personnel Data System (APDS), the DMS will primarily be designed so that system continuity can be maintained with a minimum of effort. It will also be designed to permit more responsive adaptation to changing functional requirements. The following narrative defines the relationship between data management and a DMS and outlines the capabilities of both.

Data Management should be viewed as the combined functions of information systems management (human decision-making) and generalized systems support (software processes). The human aspect is conducted by a "data manager" who, in turn, controls the software processes or data management system. Therefore, references to data management will be all-inclusive whereas references to data management system will be restricted to software processes.

The data management objectives will be:

a. Provide information that is current, correct and appropriate relative to user requirements.

b. Provide a system that can readily respond to new and changing user information requirements and facilitate the integration of external systems.

c. Permit recognition and use of the most cost/effective techniques for providing information services.

To meet these objectives, a "data manager" will be identified who is responsible for satisfying user information requirements. He will monitor and control all the operational components of the information system. These components range from data elements and codes to the computer hardware and include such things as file organizations, job schedules and utility software. Information requirements may be satisfied in many ways short of new systems design; i.e., by re-use of existing products, by unique inquiry, by report generation and by changing a code structure or adding a data element within an existing process. How the requirements are to be satisfied will be based on the estimated worth of a response versus the estimated cost of producing that response. The data management system will function to allow the data manager to evaluate the cost of producing information and at the same time give him the ability to satisfy a valid requirement by effecting change with a minimum of delay and disruption. Such change could be immediate in the sense of altering job priorities at run-time or less immediate in the sense of re-structuring data organizations.

The data management system, therefore, must support the data management objectives. Toward that end, the data management system will provide data which is responsive to the information requirements of functional users, functional managers and system operators (i.e., the data manager). Functional users will be satisfied primarily by structured applications programs; functional managers primarily by ad hoc inquiry and exception reporting; system's managers primarily by monitoring and accounting programs. The data management system will link applications programs to required data irrespective of the data base organization; it will provide file accessing for the purpose of file maintenance and query; it will provide tables to be used for specifying logical procedures, validity checks and code translation; it will prevent unauthorized use of the system and/or components; and it will measure the frequency and duration of component use. With this general definition of terms, we established the realm in which the APDS Data Management System would operate.

To be responsive to the APDS management objectives, the DMS must be able to handle very large files (toward 7 billion total characters). It must provide easy access to this data for a variety of applications required by centralized personnel management and be able to support models and simulations with statistical data. Finally, it must be flexible enough to provide unique requests for information to support the personalized management objectives.

What then are the specific functional modules of the APDS DMS? The obvious functions are access, query and file maintenance. File definition, file generation and a report generation function should also be included. Now things become slightly gray because to effectively manage data you need an accounting system to determine its use. This is also considered to be part of the DMS. User control which includes resource use accounting, priority and access privilege monitoring is another marginal function. This is viewed as a joint function of the operating system, terminal handler and DMS. Job control is definitely in the operating system's realm, but job libarary and directory maintenance may be a DMS function.

The scope of the APDS DMS will be determined by how effectively functions obtained from other systems, from the vendor and from in-house development can be integrated. The first step was to evaluate other systems that contained some or all of the pre-defined attributes. A system that attracted our attention was DM-1.

The design specification for DM-1 [7] met most of our needs. Since we would be designing a system without knowledge of the specific hardware, we hoped the implementation efforts for DM-1 would be reasonably machine independent (one of the design specifications) and that the implementation techniques had been validated. Unfortunately,

neither was true. We used a small extract file containing 40,000 records of 448 characters each to evaluate DM-1. We learned a lot about file definition, the necessary dependence on a particular machine, and that we could not expect to use the DM-1 system. However, we hope to use many of the concepts from DM-1 that we feel are valid. One is file binding with three levels of file structure. The physical file structure is the actual storage of bits on a specific device. The logical file structure is the relationship between various elements in the file. The third, the formal file structure, is the linear set of data used by an application program. The binding concept is the process of establishing the algorithm which relates the formal file and the physical file through the logical file. The primary reason for our interest in this "data independence" concept is to have the flexibility to change the content and organization of a data base without necessarily affecting all the user programs. With file binding, changes to either data base or programs require recompilation for the binding routines rather than major reprogramming.

Another DM-1 concept is the idea of a data administrator which we have called the Data Manager. The single point of control over data standards is absolutely essential in a system the size of APDS. To support the Data Manager, an administrative-data management system is necessary. For this function during our design and development work, we have looked at MADAPS [3] which is a self-contained DMS written in JOVIAL and currently implemented on the CDC 1604/160A at Rome Air Development Center. A system of this type will be used to maintain catalogues and determine relationships between files, programs, data elements and reports. Our current data base, on cards, consists of pertinent information on 1000 different files, 600 programs, 1548 data elements and over 500 reports. Eventually, accounting data reflecting user information, data element, file and routine utilization information, and hardware and software performance data would be maintained by the Data Manager's system.

After ascertaining that DM-1 would not be a viable system in our time frame and that totally building one in-house was not practical in terms of manpower and time, we looked to the computer vendors for some capabilities. Both WWMCCS and ALS live test demonstrations were underway. Both had specified data management system requirements and each system is larger than APDS. The WWMCCS specification included everything you could ever imagine a data management system doing. ALS, on the other hand, specified a requirement closely tied to the in-house developed Central Control System. We were able to determine a basic DMS capability available from industry through discussion with several vendors. The CODASYL Systems Committee Report [5] aided significantly in confirming our beliefs and estimates. The following is our initial attempt to realistically specify Data Management Software to be vendor furnished.

The DMS is defined as a group of software processes that furnish the system manager or a user the capability to define logical and physical files, load data to those files, and access the files for the purpose of file maintenance or retrieval. Applications programs using data from those files must not be concerned with the file structure, or storage media, only the data elements required.

a. The file definition capability must include:

(1) A data management language, which will allow the user to describe the logical structure for his file. A minimum of three distinct (e.g., index sequential, hierarchical, inverted list) types of structures must be available from this language. This language must allow for the use of data names to refer to data elements. It must also allow for the specification of validation criteria (i.e., size and class) for each data element.

(2) A programmatic means to assign physical storage by device or class of device to include explicit division of data across devices. It must also permit the allocation of space adequate for expected file expansion.

(3) A means of specifying the privilege/security requirements of the file.

(4) A means of specifying null values for fields that have no value at input time.

(5) The ability to specify an accounting of the frequency of file usage to include measurements against any designated data element or elements.

b. The file generation capability must include:

(1) The ability to load the files described by the data management language from an external data source.

(2) The creation of all associated index tables and directories required by the accessing scheme.

(3) At file generation time, provide for the detection of invalid data, for the output of appropriate error messages and for the insertion of null values to replace the incorrect values.

(4) The output of a memory map showing the percent of allocated space used and file/record dimensions.

(5) Initialization of file accounting system when accounting is to be used.

c. The file accessing capability will include:

(1) The ability to bind applications programs to required data; i.e., the ability to read and write individual elements and/or records or blocks of records from a specified file or files according to an I/O request expressed with the data management language incorporated in or referenced by an application program.

(2) The accessing module will provide for authority checking to insure the requesting program/user has the appropriate data privilege.

(3) The capability to count each file/element reference and record in the accounting system.

(4) The ability to access multiple files and multiply access any single file.

d. The file maintenance capability must include the ability to:

(1) Add and delete specific elements and/or records from a file with an associated accounting of file storage use.

(2) Change values of specific elements.

(3) Update indexes and/or directories as associated file values change (i.e., dynamically) or in a batch mode subsequent to a series of data file changes.

(4) Provide notification to the operator of pending overflow of allocated file space from a file maintenance activity. It must allow for reorganization of scattered data storage ("checkerboarding") and file compression.

e. The communications terminal handler will include the ability to:

(1) Transmit data to, and receive data from, all terminal devices connected to the on-line network. This module must verify correctness of any transmission and retransmit as necessary.

(2) Provide for the queuing of input/output to the terminals.

(3) Provide initial user identification and record the number of initiations for each user ID.

f. The query definition capability will include the ability to:

(1) Name the file or files against which the query is to be processed.

(2) Specify the selection criteria by using:

(a) Element names and/or elements codes as provided at file definition time.

(b) Relational operators of:

Equal to

Not equal to

Greater than

Less than

Greater than or equal to

Less than or equal to

Representative symbols or abbreviations may be used if available on furnished keyboards.

(c) Literal value for the element referenced of appropriate class and size.

(d) Logical operators of NOT, AND and OR to create compound expressions. Order of precedence for evaluating compound expressions will be: 1st NOT, 2nd AND, 3rd OR and from left to right with parenthesis being used to change precedence.

(e) Arithmetic operators of multiply, divide, add and subtract (symbols may be used) to create arithmetic expressions. These expressions may be a combination of element names, literals and arithmetic operators. The order of precedence will be: 1st multiply and divide and 2nd add and subtract and from left to right with parenthesis being used to change precedence.

(3) Output result of query operation according to the following specifications:

(a) Designation of output device to include at least the terminal of submission (if appropriate) and a line printer at the computer central site.

(b) Optional limit on number of output records.

(c) SORT output in ascending or descending order on at least two key elements of up to 16 characters each.

(d) Designate elements to be output by element name and/or code, with appropriate output identification.

(e) Designation of format according to device capability or default to system output format of columnar form with appropriate headings.

(4) Save a syntactically correct query request and to subsequently execute it on demand.

(5) Be submitted from any input source including terminals.

Armed with these specifications, we looked at some proposed benchmarks. We are not satisfied with the benchmark tests for the data management capabilities in either ALS or WWMCCS. We feel a comprehensive evaluation of all DMS functions specified must be made in a benchmark but not necessarily as part of the timed portion. The query, access and maintenance functions in our benchmark are designed to make up a representative portion of the workload in the timed tests. The file definition and generation functions will be required as pretimed test activities.

A re-load of the generated data base will be required in the timed portion. Our required response from the vendors is very general and only slightly specified beyond capabilities known to be generally available. Whatever is furnished will serve as the foundation for the APDS DMS. We are not directing our effort toward a single source for solution. In the very near future, we plan to investigate in depth the Administrative Advanced System for IBM internal management and SC-1 Western Electric's DMS which grew out of the original DM-1 specifications and was reviewed by the CODASYL Systems Committee [5].

It appears now that our initial phase of APDS will be a combination of our existing 2.5 system redesigned to handle both officer and airman systems and whatever vendor supplied DMS capability is available. Hopefully, some existing systems will be another source for obtaining additional functions. Regardless of where the various functions originate, it will be necessary to evaluate them in terms of their performance relative to the implementation objectives. To accomplish these performance evaluations, we will start with the experience acquired in Project 2.5.

Two tools used in the 2.5 development, which we will expand upon in our design efforts, are simulation and analysis.

An example of simulation as a design tool in development of a data management system is in the evaluation of specific untested techniques used in Project 2.5. The problem of a fixed length record being the same size for both generals and second lieutenants had to be resolved.

The overhead necessary to handle variable length records was considered excessive. Therefore, a scheme was chosen using a basic record with pointers indicating if there were trailer records. After simulation, this method proved to be too costly in storage accesses since most personnel applications needed all of the trailer records for any individual. Hence, the average number of accesses per individual logical record was high.

The method selected was one of creating five separate record sizes appropriate both for a basic set of data elements and the physical characteristics of the disk storage device. Using the social security number in reverse order (which makes it a more uniform random number) as the primary record locator, an individual's record could reside in any of the five physical files equally well. One logical file with basically one level of hierarchy allowed accessing to remain straight forward. The formal file in all applications was the entire 1680 characters. This technique required the performance of an elementary form of data binding (in the DM-1 sense), i.e., a transformation from the physical structure to the formal structure.

The physical structure consists of 8 characters of control information indicating among other things the presence of any of 18 possible segments. A basic set of 576 characters exists for all records and segments of variable length are appended to this set. The record sizes are 720, 960, 1200, 1400, 1680. The addition of any segment could cause overflow into the next larger record. The binding process then needs to determine only what segments are present and appropriately expand the data into the formal record.

Analysis is possible as a tool since there is an existing system and various use characteristics can be measured. An example is a study made for Project 2.5 to determine the most frequently used elements in logical inquiries. Ten of these elements were determined to be high use items in today's mode of personnel management. It was again determined and verified through simulation that these ten elements could be indexed and the data base would remain within the hardware constraints.

By inverting these ten elements, most of the logical queries can reduce the total officer file to a six thousand or less subset using only indexed items. This subset can then be processed and "same day" response provided. Counts and small subsets are provided on-line.

Data Management System Evaluation

We have a good understanding of the personnel data system and the DMS functions necessary to support it. Based upon the approach we have taken in design, DMS evaluation can be cateogirized into three separate approaches.

The first approach is shopping for an existing system. Off-the-shelf self-contained DMS's can initially be evaluated by comparison of existing implementation design constraints and the maximum design limits necessary to support your data base. If the existing implementation is such that no limits are exceeded, then tests (benchmarks) may be run directly on the system or existing performance statistics considered if a similar data base exists on the system. If, however, some limits are exceeded; then an evaluation must be made about the effect of change, magnitude of change and cost of change necessary to adjust the implemented system. Certain system design philosophies, which optimize characteristics of performance, have inherent limits associated with a hardware configuration. By changing the limits, severe changes in performance may occur. The basic system concept should be evaluated for implementation using appropriate parameters on another hardware configuration. As mentioned before, we used this approach with respect to our investigation of DM-1. The most serious deficiencies in this effort were the lack of documentation on the implementation techniques and constraints and since the implementation was still in progress, we were not able to evaluate all DMS functions individually.

The second approach is directed toward an analysis of vendor furnished DMS functions. Again a function by function matching of requirements with capability and constraints is necessary. After considering the functions individually, the effects of integrating them for a specific purpose must be determined. Thus far, our efforts in this area have been subjective and academic in nature. A real concern is the relationship of generalized DMS functions to the excess overhead necessary for generalization. The cost for marginal flexibility is an issue which must be resolved if the system cannot be specifically tailored. The supply system handling financial transactions, a personnel system and a library information system have a majority of characteristics in common. Each, however, has some unique attributes. The generalized DMS will require tailoring to meet the specific system needs. For example, a severe problem area seems to be the apparent difficulty in design of a system which can handle various sizes of data bases with appropriate efficiency; i.e., some systems which handle small data bases are incapable of handling large data bases or systems which handle large ones appear very unresponsive with smaller ones. The concept of general data base management functions seems valid, but we have serious doubt that it is possible to suggest even a general data base management system for personnel data systems. Advertized general systems seem to be general over a very small portion of the data base management spectrum.

The third approach to evaluation is the study of techniques and the selection of hardware and software capabilities, combined with these techniques to create the DMS functions necessary for a particular data system. This approach uses the tools of analysis, modeling and

simulation as we used them in Project 2.5 and expanded to cover all DMS functions. This level of evaluation focuses on the details of DMS functions. Hence, as techniques are combined to form a DMS function, the composit must be evaluated to validate it as a complete function. A function by function analysis then is made for the entire DMS. The problem of validating models and simulations adds an additional expense to the evaluation process. Also when hardware is unknown, as in our case, this approach is complicated by the need to evaluate techniques with respect to various hardware.

It is necessary for us to work in each of these three areas to combine features from our current system, an unknown vendor and any features which may be transferrable from other existing systems. The problem is one of selecting the most appropriate techniques which can be implemented on any machine within the time constraint (between machine selection and system start). This tends to force design in a high level language or in detailed flow charts. Since well defined interfaces between applications program and DMS functions are not available in this environment, continued evaluations along the lines described above are essential.

To be able to evaluate and make objective and subjective judgments with some degree of confidence, better evaluation tools are necessary.

Too often Data Management Systems are described as being "effective with large files," but "large" is undefined. Typically, the file has a maximum of 10,000 records of 80 to 400 characters. The problems of "large" files in APDS is not simply a question of more storage. We have an intuitive feeling that there are some definite step points in the efficiency of data management techniques which dictate a change in technique as the data base increases in size. These step points would be the places where the efficiency curves of various techniques intersect. In the trivial sense, a point is reached in a system where it is more practical to SORT and use a binary search than to use a sequential search. Unfortunately, most people dealing with very large data bases and files are overtaxed with their primary problem of data management and are unable to do theory and technique validation.

Data use must also be considered in the selection of a data management technique and the evaluation of system efficiency. Many systems are sold on the basis of their data retrieval performance (e.g., WWMCCS LTD emphasized retrieval). The analysis of data base activity in a personnel data system indicate most of the action is in file maintenance and report generation.

To optimize storage relative to use, the system designer must consider and understand the variables of storage expansion and access speed, defined as elapsed time from the end of a request interpretation to data

delivery to a using function (e.g., report program), as they relate to a logical and physical storage structure. Some formulas have been presented for computing overhead associated with various logical file structuring [2]. Both event and storage overhead were considered. However, sufficient data on the effectiveness of these formulas has not been accumulated to validate them. Evaluation of alternative data management techniques for minor systems or general subsystems can be made with unique GPSS simulations. A consistent but general hardware representation can be used for the comparative evaluation. Using simulation results, a reasonable judgment can be made with regard to any additional evaluation. All of the environmental constraints, machine and programming, should be included in the GPSS model. The impact on the current system can be evaluated using the SCERT Simulation System if a suitable parametric model set has been validated. The assumption is made that a valid statistical base exists from which verification of the models can be made.

There are some well formed mathematical formulas for determining average (best case and worst case) times for sort, merge or search operations based upon a particular technique [1]. The solution of these formulas for a particular situation also requires accurate statistics on the use of the specific file and data element.

Future DMS's should include implementation planning factors of system overhead requirements and performance characteristics (expressed in terms of time, accesses or space relative to the number of fields, number of records, size of record, hardware configuration, etc.) associated with the functions of file definition, generation, access, and the various maintenance tasks. Since most structuring techniques are very closely related to the use of the data, it is necessary to be able to correctly evaluate use. This suggests the need for a flexible and comprehensive accounting system associated with all the DMS functions. Ultimately, what we would like to see developed by those with implemented systems or developing new systems is a series of charts similar to those used in other engineering performance manuals. The airplane performance charts in the back of the flight manual is an example. Given temperature, dew point, field elevation, runway length and airplane weight, you can determine the expected performance. The same situation is true in using a DMS. For example, a hypothetical chart might provide expected storage requirements, load/generation time and average access time given the file characteristics (record size, number of fields, etc) some indicator of value uniqueness, structure variables and a fixed implementation (hardware). The benefits derived from this form of information is not just the initial selection of file structure, but as management policies and data use change, different structuring can be invoked to improve performance. Continuous monitoring and evaluation of the DMS functions should then give the data manager the capability to determine the impact of proposed modifications as well as adjust to changing data use factors in keeping his system tuned.

SUMMARY

Note! As promised, we have offered no conclusions. However, there are many concepts in various stages of development which will remain under consideration. Today's system designer is faced with the problem of selecting the appropriate functions and implementation techniques. He then must merge and/or tailor these to his present needs and be flexible enough to accommodate future changes.

A primary tool the designer and data manager should have is a system for measuring data use coupled with the techniques for its analysis. This tool would provide a degree of flexibility in the data management system to allow efficient response to changes in data use, personnel management requirements and other associated applications. The field of computer system performance evaluation is vast and relatively untouched. That area of DMS evaluation, particularly interesting to us, is now at a stage where meaningful research needs to be performed.

REFERENCES

1. Meadow, Charles T., "The Analysis of Information Systems, A Programmer's Introduction to Information Retrieval", John Wiley & Sons, Inc.

2. Pan, George S., "The Characterization of Data Management Systems", Data Management June 1971.

3. _____, "MADAPS USERS MANUAL" Publication of Systems Development Corporation (IM Series) July 1971.

4. _____, "Advanced Personnel Data System Plan" Publication of Directorate of Personnel Data Systems USAF Military Personnel Center, September 1971.

5. _____, "Feature Analysis of Generalized Data Base Management Systems", CODASYL Systems Committee May 1971.

6. Dodd, George C., "Elements of Data Management Systems" Computing Surveys Vol 1 No 2, ACM June 1969.

7. "Reliability Central Automatic Data Processing Subsystem (DM-1)" Design Specification Report of Auerbach Corporation (RADC-TR-66-474 Vol I) August 1966.

8. Pannas, D. L. "Information Distribution Aspects of Design Methodology" Publication of Carengie-Mellon University Feb 1971.

Air Force Online Data System (AFOLDS)

by
Major Robert A. Yorks
Captain Cecil E. Martin
Air Force Data Systems Design Center
Gunter AFB, AL

I. HISTORY

A. Background

The advent of the Base Level Data Automation Standardization Program (BLDASP) computer, Burroughs 3500 (B3500), into the Air Force gave the base environment its first opportunity to use a third generation online system. With this type of computer, all functional areas were able to receive real time response by sharing the same computer. The Base Level Military Personnel System (BLMPS) was the first of real time systems to be implemented followed by Accounting & Finance and Base Engineering Automated Management Systems (BEAMS). In the future, Civilian Personnel Management Information System (CPMIS) and Maintenance Management Information and Control System (MMICS) will be implemented. Moreover, other systems are in the planning stages.

A real time system on the B3500 computer requires three major types of programs. First, the real time system needs a master control program (MCP) with a data communication option. Second, the real time system needs an Air Force provided data communication handler (DCH) program to serve as the interface between the remotes and the functional system analyzer (FSA) programs. Third, it needs an FSA which performs the processing of messages for each functional area with an online requirement.

In the current environment, the DCH receives a message from a remote terminal which is located in a functional area and requests the MCP to roll the associated functional FSA into core. Then the DCH passes the message to the FSA. The FSA then calls the appropriate overlay for processing the message. After the message is processed, the output response is sent back to the terminal via the DCH and MCP.

B. Problem

The problem in the current real time environment is one which results from the core configuration of the Burroughs B3500 and the number of FSAs and batch programs which are to be run together. At a normal B-level base, the B3500 has 150 KBs of core. The real time processing mode would necessitate that the resident portion of the MCP, the DCH, and at least one batch program be in core. Because of this, only two FSAs, each 32KB, can be resident at any one time, but most bases have three of

the planned five systems now online. This means that core can contain FSA 1 and FSA 2 while FSA 3 is on disk. If the DCH receives an input message for FSA 3, the DCH requests that the MCP "roll out" either FSA 1 or FSA 2 and "roll in" FSA 3. This rolling in and out of 32KB programs adds substantially to the overhead as well as "non-productive" disk channel utilization. The probability of the "needed FSA" not residing in core when an input message arrives increases as new FSAs come into being. Because of the smaller core at A-level bases, the problem is compounded. Thus, if the current condition were permitted to exist, there would be either a degradation of response time or a continual request for expensive core increases.

C. Problem Analysis

An analysis of the current environment did in fact show that an increase in overhead time results with an increase in the number of FSAs. However, the analysis revealed an additional factor which led to the concept of AFOLDS. It was that there is a great deal of similarity in processing among FSA programs. These similarities were in data management functions. That is, each FSA reads and writes files, receives and transmits data to and from the DCH. Additionally, each FSA updates files, albeit using differing edit criteria, processes inquiries (information requests from a single record) on a real time basis, and produced both reports and retrievals (information requests from many records) on a delayed or batch basis. The results of this analysis and many other factors contributed to the decision to develop AFOLDS.

II. CONCEPTS

A. Introduction

AFOLDS is a general data management system for the B3500. It contains two major online programs: remote job communicator (RJC) and task processor

The RJC is core resident and will replace the current DCH. It will perform all functions now performed by the DCH. In addition, it will monitor high speed and remote job entry (RJE) hardware.

The task processor will replace all FSAs. It contains common routines which have been generalized for use by multiple functional elements. It will treat a variety of file organizations that are totally independent of file structure at data element level. Depending upon actual workload and core size,

one or more copies of this program operates in an online mode and has the capability to interface with an offline retrieval system such as base level inquiry system (BLIS) III.

B. System Overview

There are two reference techniques for accessing data elements in APOLDS. They are data access name (DAN) and data element name (DEN). DAN is a 4 character reference with the first character associated with a file. DEN is the Air Force standard 24 character data name as depicted in AFM 300-4. Therefore, there are two modes, DAN and DEN, of inputing data through the TC521.

APOLDS utilizes a variety of input methods. The major source input will be the Burroughs terminal computer 521 (TC521). Other hardware such as AUTODIN terminals and the Burroughs 2e5 (B26e5) will be used for high volume input. This type of input along with concurrent disk (a term used in MAMS to mean remote batch) data will be loaded to a disk file, pseudo remote, in APOLDS format. After loading it to disk the RJC will treat the disk file as if it were a remote.

There are 7 types of inputs used. These are: inquiry (to query a single record), retrieval (to query multiple records), update (to update one or more data elements of a record), create (add a new record to a file), delete (delete a record from a file), message (send messages to operator or other remotes), and unique (a type of input to a functional area).

The TC521 accepts and formats inputs to the RJC. The RJC handles inputs on a first in first out basis by type without regard to functional area. Subsequently, the RJC forwards all inputs to the task processor. Except on 3-levels, the RJC can roll in another copy of the task processor when workload requires it.

The task processor will rely on a series of general data management routines to process inquiries, retrievals (interfaced with BLIS III for batch retrievals), updates, creates, and deletes. For uniques inputs, a tailored overlay for that particular requirement processes the input request. The task processor forwards the response back to the terminal via the RJC and MCP.

C. Data Management Routines

Generalized data base management systems are developed and marketed today with similar features.* However, the

* Reference, May 1971 issue of Communications of the ACM, page 349.

techniques associated with them are different. A significant point about AFOLDS is that, even though it has most features (data independence from program code) of data management systems, it has some which are unique. Specifically, AFOLDS has a formalized DAta Description language (DAD) and a Data Update Edit Language (DUEL) capability.

DAD is a non-procedural problem oriented language which gives AFOLDS its program data base independence capability. Data program independence means that the description of the data is not embedded within program code. Depending upon the degree of independence, programs can accommodate varying degrees of changes in definition and structure of that data without being modified or recompiled. In AFOLDS, a functional area user can change his data structure without changing AFOLDS programs. This is done by the user regenerating file and data descriptions using DAD statements.

The DAD translator reads a source deck of cards containing DAD statements. From these statements, detailed descriptions of a users data base is generated on disk. When programs are processed against the data base, they use the generated descriptions in the same manner in which one uses a dictionary, i.e., each data item description is looked up to determine what it is and its location. Based on the referenced description, the necessary actions, such as editing, are performed on the data item.

DUEL is a non-procedural problem oriented language used to describe any type of edit for individual or groups of data elements to AFOLDS. The language is in decision logic table format. It is easy to understand and use.

The DUEL translator reads the decision tables from cards and translates them to a code on disk. This code is interpreted when the task processor needs to edit data elements before they are updated.

With DUEL and DAD as summarized above, AFOLDS is a data management system which is user oriented.

Terminal Language

There will be two levels of commands in the language:

(1) SYSTEM - communicates with the RJC.

(2) MONITOR - communicates with the task processor.

When the operator is at the SYSTEM level the commands will be free form. Commands at this level will be as follows:

MSG (to send a message to another terminal or the SPO)

STATUS (response would be some statistics about the current system status such as number of users, etc.)

TLOAD (used to load another TC521 program from a B3500 library)

LINK (to send output to designated terminals besides the originator)

DELINK (reverse of LINK)

USERS (response is last name and terminal ID of others in the system)

ROUTE (to send output to designated terminal or device rather than originating terminal)

START (to request initiation of pseudo-remote, AUTODIN, or high-speed remote)

STOP (reverse of START)

When the operator is not using the SYSTEM level of commands, he will be using the MONITOR commands. However, he will be able to freely change from one level to another and back upon demand. Once at the MONITOR level he must choose, by actuation of a program key, the following in order: (1) MODE, (2) TRANSACTION. He may choose one of these MODES:

DAN - edit limits 4 characters in DAN identifier

DEN - edit limits 24 characters in DEN identifier

Free - there are no edits and message can be approximately 150 characters in length.

Having chosen the mode, the operator must then choose a TRANSACTION type. There are seven and will be so labeled on the program keys:

Update

Inquiry

Retrieval

Create

Delete

User Function

System Function

Now the operator is ready to begin entering transactions and as long as he does not wish to change COMMAND LEVEL, MODE or TRANSACTION, there will be no need to reselect any program keys. However, by use of the OCK keys of the TC521, the operator will be able to signal the program that he wishes to change one of the three and he will be permitted to select another choice.

Use of the user function transaction will allow the user to define unique types of operations which can then be accommodated in AFOLDS. The system function transaction has no use at this date, but it is envisioned that as the system develops and commands are required to expand AFOLDS capability this provision could be used to allow online compile and execution commands, RJE commands, and the like.

The syntax rules for the transactions commands are reasonably simple and take two major forms:

(1) (key) data element identifier/data
data element identifier/data

(2) (key) data element identifier
data element identifier

Retrieval will be Base Level Inquiry System (BLIS) required format and analyzed by the BLIS syntax program for errors. The user function transaction will be pre-defined by the user so that AFOLDS will not prescribe syntax requirements for the general case. The same is true of the system function transaction except that AFOLDS will define key words and the required parameters that must be supplied or the default values if not supplied.

AN APPROACH TO DEVELOPING A DMS TEST METHODOLOGY

by: FRANCIS P. SLIWA

ABSTRACT

The proliferation of available data management systems (DMS) has complicated the process of selecting a DMS for a particular application. Though many test and selection techniques exist, the procedures for their application are informal and more an art than a science. This paper describes an approach that RADC is taking to formalize the test and selection techniques into a DMS Test Methodology. It revolves around the application of progressively discriminatory test methods against the set of candidate DMSs in a three step process. The process involves use of numerical scoring methods based on documentation for step one, modeling and simulation in step two and benchmark problems, monitors and journaling in the third step.

KEY WORDS

DMS testing, software testing, simulation, performance measurement

INTRODUCTION

In recent years the problems of selecting data management software (DMS) have increased tremendously due to the increase in commercially available generalized (and not so generalized) data management software packages. The process of selecting the right available DMS for a particular application is a complex, costly and time consuming task, a task which often has resulted in selection of the wrong DMS anyway.

The problems just begin after large outlays of time and money are made to acquire and install the selected DMS. The decision usually made at the first trouble point is to modify the DMS, since a large investment has already been made. The procedure is repeated "n" times until a mediocre to a good system evolves, which may not look like some other candidate DMS.

Because this problem has occurred all too frequently, it was decided that a program must be started to attempt to develop a test methodology which Air Force users could utilize to better and more economically match their requirements with an available DMS. It is here we run into another problem, that of user requirements which may or may not properly reflect user needs. Since this is another whole area of study, it was

deficiency in the overall process with development of a formalized test methodology and that this formalized approach to testing will be a forcing function to improving requirements specifications. It is anticipated that this formalization will primarily be a logical ordering and rationale of available test methods and DoD functions with very little requirement for new test techniques. This assumption was made after review of the test methods currently being used.

Some of the selection and test methods commonly used today are listed below:

a. Numerical scoring of weighted parameters. This approach is exemplified by the Parametric Evaluation of Generalized Systems (PEGS) study done by Informatics, Inc. for RADC. This method allows for assignment of weighted values to each of the required system parameters and provides a methodology for calculating a score for each system which can then be summarized with allowance for subjective judgments to aid a final system selection. This approach is relatively inexpensive but is limited by the quality of the available documentation about each system. This documentation, generally available from the system developers, can be misleading.

b. Benchmark problems. Benchmarks are a mix or grouping of actual applications which represent, perhaps in abbreviated format, the universe of problems which comprise the users workload. They are executed

by the candidate systems and performance figures (typically sets of timing measurements) on each system's capabilities are collected. This type of test can be costly and time consuming especially if the number of candidate systems is large. The programs and data bases must be converted to the coding and formats of each system. Major drawbacks of this technique are that performance figures for systems may be misleading due to data structures and programming strategies which may favor some systems and are not required by the application. Also, benchmark problems often characterize the existing application environment, not the desired environment.

c. Timing and Performance Measures. This category of test tools includes all those hardware and software facilities which record time and events within the computer system. These tools fall into the following general categories:

(1) Hardware monitors to count processor, channel and device activity.

(2) Software monitors to collect system software module usage statistics including elapsed time and event tracing.

(3) Journaling to record data base and DBS activity.

These tools are often available from computer and independent vendors. They can be used to collect enormous amounts of data which must then be analyzed by various techniques. They impose varying amounts of overhead on the system with the least coming from hardware.

(4) Modeling and Simulation. This method usually involves the modeling of various modules of systems such as storage, data structuring, search algorithms, user interface and then using these modules with various combinations of assumed programs and data to simulate system activities. Some examples of this are the FOREM and FORTS models built by IBM for RADC; the Advanced Airborne Command Post Computer Performance Model, and the SCERT model. These examples represent a spectrum of model types which simulate specific DMS functions at one end and total data processing systems at the other. This points up the problems of models providing too much information about a single aspect of a DMS (FORTS & FOREM) and gross generalities about the DMS function at the other end (AABNCP CPM, SCERT).

At this point I will mention another test method which is related to DMS testing and that is Information Storage and Retrieval System testing. The primary tool for ISAR testing is development of Relevancy/Recall ratio. I mention this because it must be investigated for possible application to a DMS Test Methodology.

APPROACH

Within the context of the various methods discussed above, we are currently investigating the feasibility of ordering the methods of selecting and testing DMSs into a three step culling process. The initial step would involve the use of the numerical scoring of systems using the PIECE type methods. In a particular application, this could

eliminate all but about five to ten candidate systems which could meet the requirements. This is still too large a number to physically test in a timely and economical manner. The second step could then use a set of general functional DMS models which could simulate the various candidates and give indirect quantitative guidelines for selection of two or three systems which most closely match the requirements. The third step would then consist of utilizing a combination of benchmark tests, timing and performance measures to provide more precise quantitative indicators for the final selection process.

The specific tasks to be accomplished to carry out this culling process must begin with a definition of what we are trying to test. This definition is not to be a description of what a DMS is or should be, but rather a description of all those functions which could be included in any DMS. This will likely include some functions which computer scientists will argue are not part of a DMS, such as special search functions, but most Air Force users require these functions and they must, therefore, be included. This task can be accomplished almost in its entirety by utilizing available documents such as the CODASYL System Committee reports and various DMS surveys.

The next logical task is to further bound the problem by describing the environment in which a DMS must operate. The dependencies and interrelationships of each DMS function or group of functions to the

hardware, software and operating procedures of the data processing facility must be defined. In addition the relationships among DMS functions must be defined in order to isolate the independent modules which can be logically tested. A further analysis must be made to determine which functions of this set are physically testable.

As a parallel exercise, a comprehensive list of the test and measurement facilities available in the various test and selection environments must be compiled. For example, a G635 has an operating system feature called TRACE, which at the option of the user may collect data about 64 system activities including the time of day to 1/64 of a millisecond. The GCOS accounting data is also available with information about processor, channel and device usage statistics. A similar facility exists on IBM System 360 in the System Management Facilities (SMF) option. Using SMF exit facilities, data collection routines can be inserted to determine each job step's use of CPU, I/O and storage. In addition a number of commercially available measurement products can be acquired for the 8360. These include CUE, PPE, SAM and others. The point to be made about these measurement tools is that they all test at a system level (not at DMS level) and produce large volumes of data which may or may not be useful in the DMS area. One possible area of utility in the context of DMS testing could be the use of these data to establish a normalized reference point for subsequent tests. Then, knowing the reference data and keeping all

other things constant, the DMS can be varied (e.g., vary data structures or search strategies) and the net changes can be observed. This type of test would normally be performed only in a research environment or after a system is installed to improve efficiency.

This list of measurement tools must then be added to other tests which are less dependent on the operating environment (e.g., benchmark tests) to result in a compilation of all available test methods.

The compiled list of tests must then be matched against the testable DMS functions. This involves an analysis of the measurability of the various combinations of DMS functions. Then, a proper ordering of the test sequence must be determined. For example, the execution of acceptance type tests before performance and comparison tests. A procedure to determine if a system does indeed perform all the functions claimed in the documentation. The acceptance tests validate the operability of the DMS functions and establish the known quantities for a test case. The performance tests must then be executed to determine how well the functions are performed.

An approach being investigated to solve the problem of matching tests with functions is to order the various test techniques according to the type of results they produce. For example, benchmarks produce "hard" relatively easy to use data, monitors produce much data requiring further manipulation before analysis, and subjective data from documentation reviews may be labeled as "soft" data. All the tests and DMS functions would then be arrayed in a multidimensional matrix to point out areas of test coverage, overlap, and duplication.

Another arrayed grouping which could be labeled "cost" would rank the cost of each test method. This would include manpower and facilities costs for:

- Initial startup
- Maintenance
- Set-up per measurement run
- Data Analysis
- Short-term versus long-term

Under this heading, we might expect to find benchmark costs lower for a one-shot analysis and simulation lower in cost for extended or repetitive tests.

Once the EW functions and test methods have been grouped in these arrays, it will be possible to select, based on requirements and cost constraints, a group of test techniques best suited to address the given EW test and selection task.

The tasks to this point will be done in a generalized manner for a variety of application orientations. The arrays and matrix groups will be the tools to be included in a test methodology handbook.

The next task involves the development of a specific test scenario using the groups of selected test techniques. The specific tests to be executed will then be implemented. A goal will be to have a set of 'universal' benchmarks to facilitate implementation and transferability and reduce overall costs.

VALIDATION OF APPROACH

In order to establish a level of confidence of the proposed test methodology and to demonstrate its application, it is necessary to perform validation tests using a spectrum of [illegible] and computer types. A number of scenarios will be performed on [illegible] second generation computer and a third generation [illegible] computer environment.

The current plan calls for utilization of the following systems:

a. [illegible] Data Processing System ([illegible]) on the [illegible] computer.

The [illegible] system is a relatively sophisticated [illegible] provid[illegible] file access [illegible] access [illegible] through [illegible] terminal[illegible]. [illegible] for this project, it is completely self-contained [illegible] [illegible] computer. There is no external [illegible] resource [illegible] other [illegible]. [illegible] from the [illegible] relatively sparse. A [illegible] provide [illegible] information and confidence [illegible] [illegible] third generation [illegible].

b. [illegible]

The [illegible] system [illegible] operating system on the [illegible]. It provide[illegible] [illegible] of queries through [illegible] terminals. The [illegible]

magnetic tape. This relatively unsophisticated system provides an important intermediate vehicle for understanding testing under a sophisticated operating system.

c. Sophisticated DMS on G635

The testing under (a) and (b) above should provide sufficient understanding of test methods to facilitate a smooth transition to testing sophisticated DMSs which allow complex file structures and search strategies. The DMSs to be used in the phase will be Data Manager-1 (DM-1) and/or Integrated Data Store (IDS).

The test scenario will begin with acceptance type tests to verify the operation of all the basic functions. A set of validated test data bases will be developed and maintained. The data bases will contain a variety of data types and application orientations. Next, the performance test specifications will be formalized. Each data base will be configured in a variety of file structures to demonstrate the effect of properly and improperly designed files on test results. A formal description of test results and their formats will be developed. Data in excess of that normally collected will be specified in order to [illegible] the [illegible] between actual and [illegible] data. Also, stress tests will be [illegible] to give examples of negative data to be avoided in future tests.

A variety of data analysis techniques will be applied to the test data in order to extract as much information as possible [illegible]. From these data, generalized [illegible] will be developed for data collection and analysis procedures [illegible] practical description of test [illegible].

To increase the utility and universality of the test scenarios, each test and data base will be run on more than one DMS. One important piece of data to be derived from this will be the cost of transferability.

The test scenarios, test results and sample data bases will become a part of a data base to be used in conjunction with the test selection and design procedures to form an overall test methodology. Such a collection of data could preclude the need for some users to perform some physical tests and substituting instead the applicable data in the data base.

SIMULATION AND MODELING

The discussion of simulation and modeling has been deferred to the end because, even though it is an area of possible high payoff, it is a longer range goal to assess the utility of available models and develop necessary additional models.

Two existing simulation models, SCERT by Comress and the AMBUCP Computer Performance Model, simulate a total data processing capability including channel, CPU, peripherals, and software module operations. Because of their scope, they provide rather gross data about data management system activity. The type of detail data required for just DMS test and selection requires a more detailed model implementation approaching the FORMS and FORET type models. FORMS and FORET emphasize data structures, content, physical data organization, accessing techniques, and physical

characteristics of direct access devices, and de-emphasize operating system and other software events. Currently, both models are limited to IBM device characteristics and access techniques.

An approach being investigated is to bring these models to a higher degree of generality and supplement them with models of the other basic DBS functions.

If the input/output data of the models can be standardized, the models will be used in several configurations to model various available DBSs at a general level. This information generated by model runs will then be used for the second step of the culling process mentioned earlier.

SUMMARY

This development effort is in its early stages and is being presented to make AF users aware of its existence and elicit their comments on how a DBS Test Methodology can be tailored to their needs. We are especially interested in acquiring more user data bases, application scenarios and DBS simulation models.

Throughout this paper, the term "evaluation" has been avoided. This has been done because evaluation assumes an understanding of user requirements and an ability to form judgments about a particular system's ability to meet those requirements. A general test methodology cannot presume to make these judgments but merely provides sufficient data, and methods for analyzing these data, to assist a user in selecting the best DBS for his application.

BIBLIOGRAPHY

1. Bairstow, J. A. "A Review of Systems Evaluation Packages," Computer Decisions, (June 1970), 20-23.

2. Budd, A. E. A Method for the Evaluation of Software: General Description Including Benchmarks, Mitre Corp Tech Report Vol 1, ESD TR-66-113 Vol 1, (August 1966).

3. Campbell, D. J. and W. J. Heffner. "Measurement and Analysis of Large Operating System During System Development," Proceedings AFIPS, FJCC, Vol 33 (1968), 903-914.

4. Codasyl Systems Committee Technical Report. Survey of Generalized Data Base Management System, May 1966.

5. Codasyl Systems Committee Technical Report. Feature Analysis of Generalized Data Base Management Systems, May 1971.

6. Dorchont, A. J. et al. A Methodology for Comparison of Generalized Data Management Systems: PECS (Parametric Evaluation of Generalized Systems), Information, Inc., ESD-TR-67-2, March 1967, (AD 811682).

7. Fife, Dennis. Alternatives in Evaluation of Computer Systems, Mitre Corp MTR 413, ESD TR-67-389, December 1968.

8. Fry, J. P. et al. A Survey of Data Management Systems, Mitre Corp. MTR 5036, March 1968.

9. Kolence, K. "System Measurement: Theory and Practice," Info Tech State of the Art Report #1: The Fourth Generation, (1971), 391-406.

10. Lucas, Henry C., Jr. "Performance Evaluation and Monitoring," Computing Surveys, Vol 3, No. 3, (Sep 1971), 79-91.

11. Vander Noot, T. J. "Systems Testing - A Taboo Subject," Datamation, Vol 17, No. 22, (15 Nov 71), 60-64.

ENVIRONMENTAL TESTS

OF

ADVISOR DATA MANAGEMENT SYSTEM

ON

HONEYWELL GE/635 COMPUTER

by:

RAYMOND A. LIUZZI
JOSEPH P. CAVANO

ABSTRACT

RADC's involvement in data management has led to a number of data management testing activities. A series of environmental tests were conducted on a data management system known as ADVISOR which has been implemented at RADC and U.S. Air Force Data Services on their Honeywell GE/635 computers. A description of the Advisor data management system is given and the following four areas of the test procedures are emphasized in this paper:

1. Original strategies for testing
2. Problems encountered
3. Results
4. Recommendations for future testing

KEYWORDS

DMS testing, software testing, simulation, performance measurement

INTRODUCTION

The Rome Air Development Center has been involved in many aspects of data management for the Air Force. One of these aspects has been testing and evaluating various data management systems on the Honeywell GE/635 computer. As part of this testing activity, RADC worked with ESD, AFACS, and Mitre personnel to provide a data management capability to the Air Force Data Services Center. It was decided that an interim system would be selected to assist in their immediate data handling and data management problems This approach provided the time for the Data Services Center to study in greater detail all potential systems that might meet their long-range problems.

Advisor, a data management system developed for the National Aeronautics and Space Administration (NASA), was chosen as the interim system for Data Services. There were two main reasons for choosing Advisor. First, it executes on the Honeywell GE/635 computer system which both RADC and Data Services have. Second, Advisor was available at relatively little cost to the Air Force.

RADC's initial testing of Advisor began after copies of the system were obtained on tape and brought to RADC for implementation. Advisor was then run on the Honeywell GE/635 system in a strictly batch environment for a period of three months. It soon became apparent that easier accessibility to the system should be provided to the user. It was decided to accomplish this by implementing the query building portion of the Advisor system as a sub-system under the GECOS Time Sharing System. After this was implemented, RADC proceeded to conduct a series of environmental tests to determine the impact Advisor would have on the GECOS operating system and how various Advisor users would be affected within the GECOS environment.

BACKGROUND

A brief history of Advisor's background is needed to fully understand the system. Advisor's framework lies in three distinct data management application programs (IMS, AMIRS, and MAIDS). IMS consisted of a data edit module, a file management module, an extract (search) module, a sorting module, a computer module, and a report generator. AMIRS was basically a batch processing file generator that enables a user to build files of data and define them by means of a dictionary. MAIDS was an on-line visual display system that complements and works closely with the other programs, yet can still be thought of as a distinct data management system. Each of these distinct data management application programs was molded together under a common main control program. Rather than designing and developing a completely new information management system, Advisor thus, evolved as an outgrowth of these earlier systems.

What does all of this mean in terms of actual system usage? Briefly, this development of Advisor has produced a highly interactive, easy to use system with a wide variety of capabilities. With few interfacing problems, a number of user application programs can effectively call upon these different capabilities. Essentially by combining different DMS capabilities which complement each other, Advisor has provided selective and comprehensive information management capabilities with a minimum amount of reprogramming necessary for effective usage.

Advisor is primarily written in Common Business Oriented Language (COBOL) along with some modules in Fortran and GMAP. The modules of Advisor handle data. These individual modules are used in different combinations to provide a

complete information management capability. Each module has linking capabilities to other modules so that they can be used in conjunction with specific programs to provide a selective and comprehensive data management system. The functional modules of Advisor form a building block system which performs the normal data manipulation function of an information data management system.

The original concept of the Advisor system was to provide two separate but related data management systems under a Common Executive. These two systems are known as the Direct Access Command Edit (DACE) System and the Direct Access Data Display System (DADDS) System. The DACE System allows a user to build a query to sequentially search a data file by using a series of statements with Boolean criteria along with sorting, computing and printing parameters. A user is given the ability to formulate a comprehensive on-line interrogation against a data base in this query development mode. A user may also reformat his description of a file if he determines that it would be more advantageous to work with a new file definition. The DACE system also provides a comprehensive set of tutorial assistance messages which a user may or may not choose to use. Under the computer aided construction mode (CAI), a user is stepped through a series of questions and answers prior to developing his query. An experienced user,however, may by-pass the tutorial mode and implement his query directly. In addition, DACE provides a comprehensive set of error diaognostics designed to assist the user in developing his queries. If a user encounters a syntax or field error while inputting his query, remedial action may be necessary and DACE allows a user to correct his mistake interactively before the query is executed.

The DADDS component operates under direct access with a user having direct interaction with his data. This system essentially works with a file described in a hierarchial tree structure format with an index of nodes provided to the user to inform him of data content at a given node. This system is essentially an on-line, highly interactive, and a fast retrieval system. A user who is thoroughly familiar with his data base can obtain information very quickly using DADDS. In contrast to the portion of Advisor which is concerned with retrieving records based on specific criteria, DADDS is concerned with displaying information directly from a pre-specified record. DADDS displays the data of a given node in tabular form depending on how a user has previously described a definition of this particular display.

Essentially, RADC is concerned with both DACE and DADDS. It is the feeling that both of these systems could be made to complement each other in a useful manner. It is with this intent that RADC decided to explore the possibility of combining these systems. The following is a complete description of Advisor emphasizing the DACE and DADDS subsystems as they are currently implemented on the Honeywell/GE 635 computer.

SYSTEM DESCRIPTION

As previously stated, Advisor consists of two main subsystems - DADDS and DACE. However, the system as a whole may be divided functionally into four primary areas.

A. Advisor - Executive - Control - Function

B.. File Mainteinance Function

C. Information - Retrieval - Function (DACE)

D. Data Display Function (DADDS)

The Advisor Executive Control Function manages the internal resources of the system. It controls the input of queries, how they are stored and how they are finally run. It also manages the overlaying of modules during execution of a batch query. In addition, it monitors multiple users who can gain access to the system through the Direct Access mode of GECOS. Finally it controls both the DACE and the DADDS subsystems.

The File Maintenance Function deals with the creation of a dictionary which describes a data file to Advisor. The dictionary contains the data description of files for both DADDS and DACE. The dictionary includes a standard format for the generation of a report that may be chosen automatically in a query. These data descriptions can be updated and maintained easily. Fields may be deleted, added, or modified. In addition file maintenance of a Security File is also provided. This file limits users access to only those data files that they have permission to query. This permission is described during the development of the Secutity File.

The information retrieval function supports the main function of the DACE Software package. A query building portion of this function, known as Aids, operates under the GECOS Time Sharing system. The batch portion operates within the normal GECOS programming restraints of any batch program.

A user develops his query by interacting through the Aid system as a GECOS Time Sharing user. The system was placed under time sharing to allow rapid access by all users at any time. In addition to this rapid access, time-sharing allows a query to be built, saved, and finally executed from a users terminal.

There are presently two modes of operation under the Advisor Time Sharing system: The Query and Expert Modes.

The Query mode contains tutorial assistance which provides detailed instructions for the development of a query by a relatively unsophisticated user. It also provides an overview of the whole system operation in a tutorial manner. The Expert mode is for a user who is already familiar with the system, and it allows him to rapidly input a query and execute it.

A completed query once accepted can then be placed in the system in either of two ways. One way is to place it in what is known as the batch collector file. There, all the saved queries will be batched together and run at a later time. This feature allows queries that are to be executed against large data bases to be run during a period when computer resource demands have diminished on the operating system. The other, a priority run, is the immediate execution of a query in the batch world with other GECOS batch jobs. This mode of operation is used when a user requires the results of his query immediately.

The functional capabilities of DACE are separated into five modules:

1. Search
2. Sort
3. Compute
4. Print
5. Reformat

The Search Module provides the functional capability to read and selectively retrieve records from a master file and pass them on to other batch modules for further processing. This retrieval is accomplished by using Boolean Criteria to compare either specified fields against user inputted values or one data field in the file against another file. The Sort Module permits records to be reordered on the basis of values found in selected data fields and may be done either in an ascending or descending sequence. The Comput Module provides the capability of calculating on the basis of user inputted formulas using the data extracted by the search module. The calculation formulas are simple Fortran-type statements and include Addition, Subtraction, Multiplication, Division, Exponentiation, and Summing capabilities. Furthermore, the Compute Module has the ability to do these calculations depending upon changes of values in specified fields. The Print Module contains a limited report generation capability. The Print Module displays the data either in the form originally specified in the file dictionary or in a new format specified by the user during the input of a query. The basic capability is a tabular display of data which allows for data suppression based on changes in the field, and a table look-up feature which translates coded values of fields into meaningful information. This saves space in the data base by having one-character codes replacing large amounts of data. Finally, the Reformat Module enables users to take data from a file, reorganize it in a different format and place it on a new output file.

Once a user has formulated his query under the time sharing subsystem he can then execute it utilizing the batch subsystem of Advisor. This subsystem executes in a normal GECOS environment and must compete for the GECOS resources needed for allocation. These batch jobs normally execute in 34K core with a minimum of 1 tape drive which contains a data file. The batch subsystem may execute one priority request or a number

of requests that have been placed in the batch collector file. Each of the queries successfully executed under the batch system will generate an output report. This can either be printed at the computer site using a high-speed printer or directed to a permanent file which may be scanned and printed by a user at his teletype.

The DADDS subsystem operates under GERTS and may directly complement the DACE subsystem. While the DACE portion is concerned with retrieving records based on user specified criteria, DADDS is concerned with retrieving data directly from a record. Basically, DACE produces all the records from a data base where the values of certain fields meet certain requirements. DADDS, on the other hand, displays on-line information in a specified record regardless of field values. DADDS is an on-line visual display system that enables a user to access and modify previously stored information. Data is stored in a tree-structured file, and any node or any branch of the tree may be accessed directly.

DADDS CONTAIN THE FOLLOWING FUNCTIONAL CAPABILITIES:

1. Subsystem Control Command

The Control Command permits users to directly access any node in the DADDS tree-structure. This display also allows the user to browse through an index containing the names of nodes at any branch or at any level of the file and shows how they are related. This module controls the inputting of definition and display formats and also contains catalogues of these same definitions and displays.

2. List Displays

The list module retrieves the contents of a specified record at a certain node and displays the information in tabular form with variable column positioning and line spacing. The user may also specify titles and column headings for the displayed data.

3. Data Plotting

The Plot Module generates numerical data from a DADDS data base and plots up to three dependent variables in graphical form for the on-line user. The user specifies the variable fields that are to be used, the nodes where they are to be taken from, and the labels which will describe the plot. Plots may be chosen to be either non-cumulative or cumulative.

4. Data Update

Updating the value of any field may be done in an on-line environment. A permanent update changes the field value permanently. A temporary update changes the value only as long as the user is on-line. When he signs off, the original value is reinstated. In any update, the modification is displayed back to the user who must approve it before it can take effect.

5. Problem Identification

Although searching the file by normal Boolean criteria is not available, a Problem Identification module exists which compares the contents of user-specified numeric or date fields against the contents of a user specified reference field to determine if the data in those fields exceed the others by more than a pre-specified amount.

6. Data Summarization

The summary module provides users with the capability of adding the contents of specified fields within the data base. The fields may be summarized either in the same node or across nodes. The summarized result is displayed in an optional format.

Design of Environmental Tests

In designing the environmental testing procedures, RADC personnel decided to investigate what effects various load parameters would have on the system. These parameters included:

1. Number of ADVISOR users
2. Types of Queries Generated
3. Core Size of Time Sharing Module
4. Types of TSS subsystem users and others.

In addition to these external parameters, the environment of the Honeywell GE/635 Computer at a given instant of time was considered critical. That is, the GECOS operating system can service local batch, remote batch and time sharing users. Jobs executing in the batch mode normally range from 20K to 100K core. These jobs may be heavily processor bound or I/O bound or a combination of both. Under the time sharing system, the user load may vary from 20 to 55 users each of whom has access to a number of subsystems such as Fortran, Basic, Cardin, etc.

In designing, the experiment, it was felt important to try to determine the job mix in execution at a given moment along with the time sharing load mix at the same time. If in fact these statistics could be substantiated at a given point in time, then logically the affect on these parameters due to the presence of ADVISOR users could be ascertained. Therefore it was decided to collect data in both these areas. The GECOS operating system provides a daily accounting report of all activities within the system. This report contains statistics on activity type, core size, processor utilization, lines of output etc. In addition, a Time Sharing Usage Report can be generated by the master time sharing user. This report contains statistics of number of users, subsystem usage, time sharing processor

utilization, etc. With this data, it was hoped that a determination of actual system utilization at any given point in time could be ascertained.

Furthermore, in designing the environmental test , it was felt that the number of users on the time sharing module would have a definite effect on response time. In conjunction with the number of users, the amount of core allocation and types of time sharing user should also effect response times. That is, if time sharing is heavily saturated with large subsystem users (i.e. FORTRAN, BASIC),response times should in general increase. If the system is saturated with smaller subsystems users (i.e., CARDIN, SCAN) response times should be rapid. In addition wanted to determine how ADVISOR users would affect this configuration. To complicate the situation, we felt we should be prepared to analyze these effects under varying ADVISOR users. That is, as we increase the number of AIDS users, what effect do we have on other time sharing users?

In our analysis of an AIDS response time, we have had, in effect, to determine what is meant by an acceptable response time. Obviously, each person is likely to have his own idea of what is acceptable to him. Furthermore, this acceptable time is likely to change from day to day depending upon the situation and the mood a person finds himself in. Thus, the acceptable response time that we take as our standard is done so with following considerations. It is generally accepted by many that any response in the 5 to 10 second range may be deemed instaneous. In fact, any response time less than 5 seconds may be unwanted due to the psychological problems encountered when the computer responds too fast.

People think the machine is anticipating their moves. On the other hand, waiting for more than 15 seconds tends to promote boredom and the fear that the computer has crashed. Based on these considerations, we have thus set 15 seconds as roughly the upper limit to an acceptable response time for our initial testing phase.

The integration of all these parameters into a suitable test plan is one of the goals of this environmental testing program. However, with the considerable number of variables present, the probability of achieving the desired controlled atmosphere for testing on the GECOS system is very small.

On the further design of the environmental tests and as actual testing commenced, it became readily apparent that because of the many parameters and complexity of controlling the GECOS operating system for any length of time, a pseudo-controlled atmosphere must be attempted. Therefore, it was decided to monitor one or two time sharing users on a daily basis under reoccuring circumstances. In addition, these users were to continue their same actions while AIDS users were exercising the system. Hopefully, these users would notice any effects that could be attributed to the addition of AID subsystem users.

To attempt control of ADVISOR usage, it was decided to determine a set of queries which would exercise all functional portions of the system. These queries would be pre=punched on paper tape for input via a teletype. It was hoped that by continually exercising the system with these same queries, but with a varying user load, a noticeable effect in response time would be reflected. If such effects occur, it was felt a "threshold" could be established at the point of concern.

We have thus set the goals in the design of the environmental test. First is to determine how the AIDS subsystem reacts in the GECOS time sharing atmosphere and how it affect other subsystem users. Secondly how does the presence of the whole ADVISOR system within the GECOS operating system affect its performance and vice versa.

Testing Results

During the actual testing of ADVISOR, it soon became readily apparent that because of the large number of parameters involved, it would be very difficult to achieve the controlled atmosphere that was originally planned. The results of the test and a graph indicating an acceptable response range are contained in appendicies A and B. A sample of each query is also indicated.

In observing the results, one can find some interesting points. First once a query is input, acceptance consists of an input phase during which the paper tape is read by the AIDS subsystem and then stored on a collector file. It became readily apparent that actual storing of the query on a collector file was contributing to the heavily responst time of the completed query. In fact, the mean response times showed more than a doubling in size when responses included the final acceptance of the query. As seen in Table 1, not only did the mean response time increase when acceptance of a query was included, but the standard deviation also increased by a greater rate. In fact, it was not uncommon that this single response time was more than the time needed to input the whole query prior to writing it on the collector file.

From the time sharing usage reports, it can be ascertained that as the number of time sharing users increase the amount of file I/O also increases. This suggests that as the number of users on time sharing keeps building up, larger subsystems will find it increasingly more difficult to complete any file I/O. This, in fact, could explain why storing a completed query on a collector file requires excessive time. First when a user inputs a query to AIDS, the amount of file I/O necessary is to a buffer area within the immediate accessibility of the subsystem. However when the AIDS subsystem

attempts to write the completed query on a collector file, it must resort to a permanent file outside of its immediate control and thus must compete with all other large subsystems for the necessary file access. This is turn results in additional time.

The second result discovered concerned the relation of the response time recorded in the testing activity versus the total number of TSS users. From the graph, we have noted what we feel an acceptable response range is. Now we can see that the mean response times for 31 or less TSS users were almost always in the acceptable range, while the times for periods when TSS users equaled or exceeded 33 users were in an unacceptable range. If we also consider the fact that for each of these points, the time sharing core size varied from between 60 and 80K, an even more remarkable result is indicated. That is, even with additional core allocated to time sharing, degradation of TSS user response times were recorded. This suggests that there may exist a "threshold effect" regarding the number of time sharing users and response times. That threshold appears to be at 32 users.

It should be noted that the limited data collected does not clearly extablish the fact that this "threshold" exists but does point out that one may exist. In addition, this result can further be applied to any large TSS subsystem. That is, if a "threshold" does exist, and there may be more than one, this "threshold, should be applicable to users of both the FORTRAN and BASIC subsystems. That is the AIDS subsystem is essentially the same size as these other subsystems and their behavior under time saring should be similar.

A third result of the experiment was the fact that the presence of the AIDS subsystems did not seem to seriously effect other time sharing subsystems. That is, the small subset of TSS users assigned to running daily repetitive tasks under various time sharing subsystems, exclusive of the AIDS users, were not seriously affected in their response times when AIDS usage increased, This suggests that the presence of the AIDS subsystem affects the operation of GECOS only by increasing the number of time sharing users.

Other testing results indicate that once the "threshold" of 32 users is exceeded, it becomes very demoralizing and dissatisfying to an AID user to attempt to input a query. That is during this period of "threshold exceedance" it might be wise to limit AIDS usage. If during this period it is necessary to utilize ADVISOR, the priority within the time sharing system of the AIDS subsystem could be raised. This alternative was not attempted during this series of tests.

Due to lack of documentation on the DADDS subsystem, a thorough analysis of its functional capabilities could not be made. Most of the testing on this subsystem was directed at just getting the system operational with a known data base. Nevertheless, we were able to verify the existence of functional capabilities of at least four modules. These included the subsystem control command, the list displays, the data update, and data summarization. The problem identification and data plotting modules were either unworkable or not fully debugged and did not achieve their full potential.

However, the ease at which a user could persue his data base through the use of an index and data display function inidcated that the usefulness of the DADDS subsystem was certainly worth further testing. It definitely seemed feasible that both the DACE and DADDS modules could complement each other in some workable manner.

An additional result that should be stressed was the fact that regardless of the amount of data generated, it is very difficult to determine the state of the operating system at a given time. That is even with accounting reports and time sharing usage reports, a picture of actual system usage at an instant in time is difficult to construct.

Also, the amount of data collected in relation to the amount of data actually used was not consistent. That is, even with the inordinate amount of data collected and vehicles for collecting more data available, we were not able to fully utilize it. Therefore, this suggests that in further testing, we should be more careful in data collection procedures. The amount of data collected does not seem to be a criteria for evaluation but rather the type of data and how it fits the objectives of the testing program is important.

Observing the results of this testing program, we cannot help but feel the need for a more thorough and vigorious testing of data management systems. The testing of the ADVISOR System has pointed out how difficult it is to conduct a controlled experiment and obtain valid data. With the constantly changing environment of a large operating system the ability to set up and control a testing atmosphere becomes almost impossible. However,the fact that results and data are obtainable about system performance, suggests the need to continue to extensively develop techniques for future testing.

As far as what we actually learned in this testing activity, we can relate this analysis:

a. Objective. The objective that was originally set was too ambitious, and too general in nature. We feel more care should be taken to develop precise objectives which can be clearly defined. The objective for out testing activity should be attainable through a series of approaches that are realistic and achieveable.

b. Data Collection . Although data was collected in a number of areas, it was much too difficult to fully utilize all of it. The form that the data was delivered to us did not readily lend it to an analysis. This suggests that in further data collection techniques, a need may exist to massage the data prior to analysis or software programs should be written to collect only pertinent and specified data.

c. Data Correlation . The experiment was conducted over a long period of time which made it difficult to coordinate all the data collected. In fact, the changing status of the operating system makes it difficult to determine if in fact, data collected on different days has any correlation.

d. Controlling Operations . Our original goal of controlling a large number of parameters soon become unmanageable. We had difficulty just in getting people to input paper tapes via a teletype. During actual testing, machine problems caused a disruption of the planned activities. The "logistics" of just coordinating people and machines to conduct a meaningful test became a real problem.

From the results generated, a cursory examination of the response times generated suggests that a "threshold" of response times for time sharing users may exist. Further analysis could suggest another threshold at other numbers of TSS users. In fact, there may exist some step function where responses remain relatively constant at each level of usage, but jump drastically at the indicated thresholds. Carrying this concept further would be of interest in further testing activities.

As far as the complete testing activity, we feel the most important result was the learning process involved. Although many techniques for testing have been suggested, the fact that we were actually able to generate and analyze data is important. However, we feel a great deal more work needs to be done in this area; especially regarding data management systems.

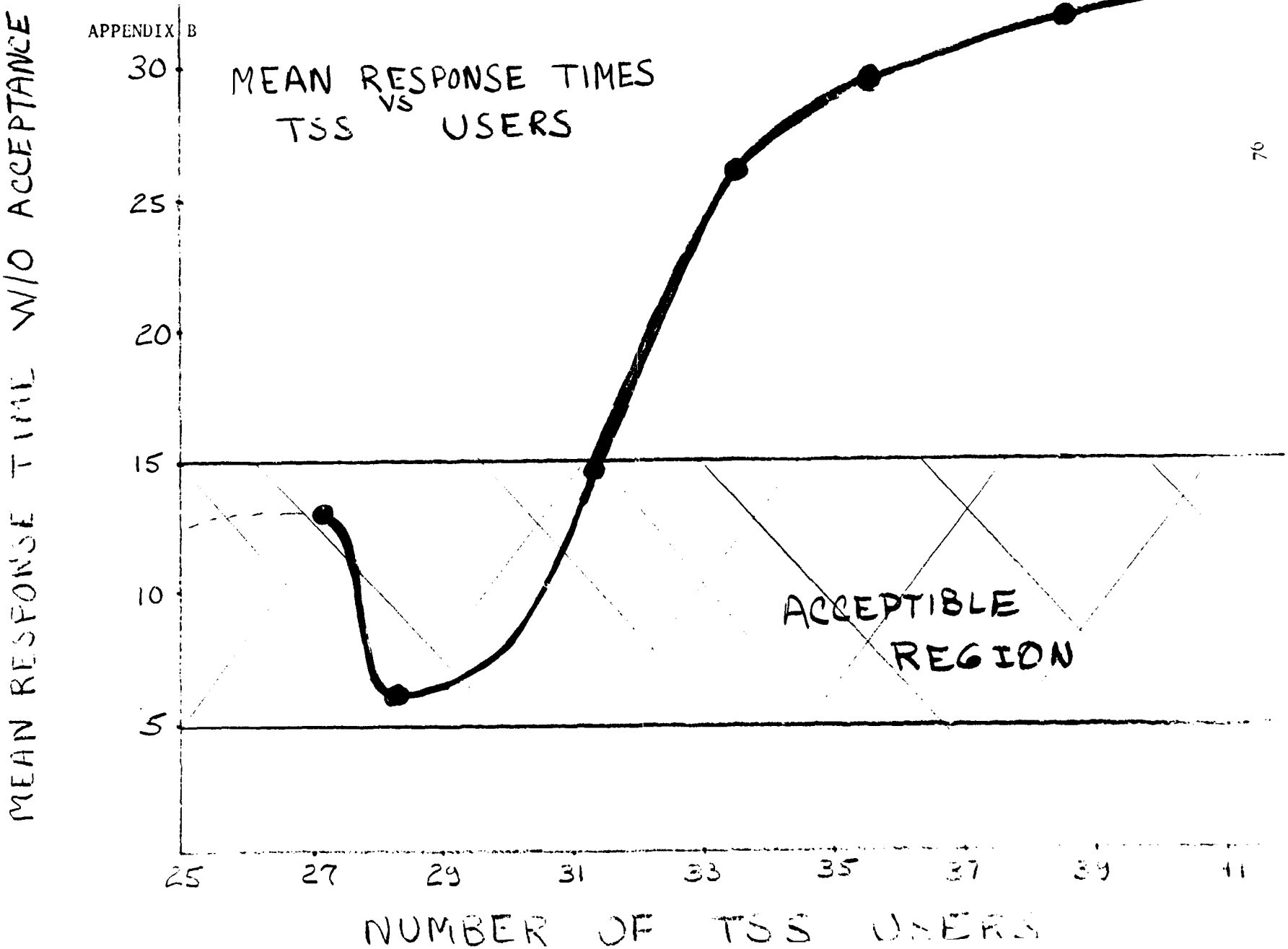
Some of our analysis suggests that we may have to utilize other techniques in future testing activities. To aid in controlling parameters, simulation may be necessary. Within this testing phase, it has become apparent that a need definitely exists to model various data management functions using simulation techniques. In fact, some of the results gained from this environmental testing could be used to validate simulation models.

Therefore, in future testing, we can hope to utilize the results obtained from operational testing, combine them with tools from simulation concepts and hopefully further develop a methodology for testing techniques.

APPENDIX A

Number of Users	Without Acceptance		With Acceptance	
	Mean Response Time Sec.	Standard Deviation	Mean Response Time Sec.	Standard Deviation
27 (60K)	13.4	9.7	27.	59.5
28 (60K)	8.0	7.3	26.6	61.7
28 (60K)	5.8	3.7	11.1	20.1
20 Average	6.7	1.1	17.2	7.2
31 (60K)	16.0	14.4	19.7	20.0
31 (60K)	9.8	10.3	17.5	34.0
31 Average	13.7	3.0	18.9	13.7
33 (80K)	26.0	10.2	61	
35 (80K)	29.0	8.0	75	
38 (80K)	32.9	31.3	94	

APPENDIX B


MEAN RESPONSE TIMES
TSS VS USERS
MEAN RESPONSE TIME W/O ACCEPTANCE
30
25
20
15
10
5
ACCEPTIBLE
REGION
25
27
29
31
33
35
37
39
41
NUMBER OF TSS USERS

BIBLIOGRAPHY

1. Vander Noot, T. J. "Systems Testing - A Taboo Subject," Datamation (15 Nov 71).

2. Marcus, C. A. "Aids Users Manual," (15 August 71) MTR-2203 Mitre Technical Report.

3. Lucas, Henry C., Jr. "Performance Evaluation and Monitoring," ACM Computing Surveys, Vol 3, No. 3 (Sep 1971).

4. Mac Dougal, M. H. "Computer System Simulation An Introduction," Vol 2, No. 3 (Sep 1970).

5. Nimith, Alan G. and Rovner, Paul D. "User Program Measurement on a Time - Shared Environment," Communications of ACM (Oct 1971).

6. "Manned Space Flight Data Processing System," MA001-009-1 Volume 1 Parts 1-4 National, Aeronautics and Space Administration.

7. Honeywell/GE Comprehensive Operating Supervisor (GECOS-III Honeywell Information System Inc. CPB-1518B (Oct 69).

The MITRE Corporation
Bedford, Massachusetts

AN APPROACH TO THE EVALUATION AND SELECTION OF DATA MANAGEMENT SYSTEMS

Barry L. Gerken
Command and Management Systems

Sponsorship - Air Force Contract Number F19628-71-C-0002

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

Since the early 1960s, The MITRE Corporation, under the sponsorship of Air Force Electronic Systems Division (ESD), has been involved in designing, implementing, testing, evaluating, and selecting data management systems. During this period, the government and private industry have made a substantial investment in data management system technology. A full return on this investment will not be realized until data management systems can be acquired by all who need them. Two major obstacles to this acquisition are the determination and documentation of requirements for a system, and the subsequent evaluation and selection of a system. The Command and Management Systems Department at MITRE is currently involved in a pragmatic approach to reducing these obstacles as part of the ESD Data Management Systems development program. One effort is identifying potential users of data management systems within the Air Force, and developing guidelines for the collection of requirements from such users. Another effort is developing a systematic approach for evaluating data management systems based on the types of file handling problems which are most suited to particular classes of systems. Practical experience gained by running full-size file handling problems on several existing data management systems is the basis of the evaluation effort. This paper describes an approach to the evaluation and selection of data management systems which incorporates the knowledge and experiences gained from past and current activities.

CONTEXT

The term "data management system", as used in this paper, refers to traditional generalized data management systems.[1] The distinction between self-contained systems and host-language systems is not precise and is best illustrated by example - NIPS/FFS, MARK IV, GIS, and TDMS are examples of self-contained systems; IDS and IMS are examples of host-language systems.[1] A self-contained data management system, or DMS, is assumed to incorporate some or all of the following functions: input editing and validation, file generation, file updating, file maintenance, selective retrieval from files, and editing and formatting of output.

The purpose of the evaluation and selection process described here is to acquire the best DMS for a particular application or set of tasks. The approach is not intended for a generalized comparison or evaluation of DMSs independent of specific applications. This paper describes the steps the Air Force should follow in order to select DMSs for immediate use within installations or organizations.

MEANS OF ACQUIRING A DMS

A DMS may be acquired in three ways: obtain an existing DMS and use it as is, obtain an existing DMS and modify it to fit specific needs, or build a new DMS. Obtain usually means buy, although some DMSs are nominally free within the government inventory. The three options must be carefully considered to determine the most efficient and least costly means of fulfilling the need for a DMS.

The relative costs of the three options for obtaining a DMS are shown on Table I. Existing DMSs may be purchased for prices ranging from $25 to over $100,000. Costs for the "buy and modify" option are estimated for a modification causing little or no change to program logic, such as converting a system from one type of computer to a different type of computer (e.g., from IBM 360 to CDC 6600). Extensive modification of logic or extension of system features could be as costly as the "build" option. A commonly accepted cost of building a full-scale DMS is $3,000,000. The difference between the "buy and use as is" option and the "build" option is two orders of magnitude in dollar cost and one order of magnitude in elapsed time.

EMPHASIZE USE OF EXISTING SYSTEMS

The evaluation and selection of a DMS for the "obtain and use as is" option discussed above is stressed in this paper. Although this emphasis is completely justified on the basis of the cost and elapsed time factors (Table I) alone, other considerations reinforce this position.

Many DMSs Available for Many Hardware Classes

The number of existing software packages that perform DMS functions is very large. Most of these are proprietary software systems being marketed by commercial vendors. An August 1970 survey[2] listed nearly 100 generalized file management programs. A software reference service[3] surveys nearly 40 information storage and retrieval systems. A recent catalog[4] identifies more than 150 systems which are available for nearly all major modern data processors in the Air Force inventory. These are summarized in Table II.

Available DMSs Have Broad Range of Features and Performance

The range of features and performance of available DMSs is very broad. Many have English-like query languages. Many allow on-line query and maintenance of files. Several make use of the same file structures and access methods supported by the vendor operating system. Most provide for efficient batch file updating and maintenance.

TABLE I

COST OF OBTAINING A DMS

OPTION	DOLLAR COST	ELAPSED TIME
Buy and use as is	Tens of thousands	Weeks to months
Buy and modify	Hundreds of thousands	Months to years
Build	Millions	Years

TABLE II

SUMMARY OF AVAILABLE SOFTWARE SYSTEMS
PERFORMING DMS FUNCTIONS

Manufacturer	Computer Model(s)	Number of Available Systems
Burroughs	B2500/3500	4
	B5500/6500	8
	B6700/7700	2
Control Data	3100/3200/3300	3
	3600/3800	1
	6000 Series	6
Digital Equipment	System 10	2
Honeywell	H200/1200	10
	GE 200 Series	3
	GE 400 Series	1
	GE 600/H6000 Series	8
IBM	System 360 Series	106
RCA	Spectra 70 Series	26
Univac	U1100 Series	16
	U9000 Series	4
Xerox Data Systems	Sigma Series	4

NOTE: Some systems are available for more than one type of computer.

Performance varies considerably among systems, usually due to design and implementation tradeoffs among the file generation, updating, and retrieval functions. Some systems stress a balanced performance, e.g., moderately fast file generation and moderately fast retrieval. Other systems stress rapid retrieval, usually at the expense of time-consuming file generation and update functions. At least one system of this type offers total on-line query response time measured in seconds for files containing hundreds of thousands or millions of records. Generation time for such files would be measured in hours of processor time.

Large and Diverse Number of Users

Many existing DMSs have dozens and even hundreds of users, including most major industries and many large corporations, but also many smaller firms. Systems with many users over a period of time are likely to be well-tested and have a proven history of usefulness. Commercial applications range from maintaining and searching large files of bibliographic and abstract material for research libraries, to handling personnel files, to providing very rapid on-line response for telephone directory assistance personnel in a major city.

Success Likely

Many DMSs are available and are being broadly applied to a diverse set of problems within the commercial world; it is likely that these existing DMSs can be broadly applied within the Air Force. Certainly in the area of management support, the problems faced by Air Force managers are not likely to be greatly different from those faced by managers at major corporations. Both types of managers are dealing with large volumes of data describing resources - - their personnel and the jobs they are doing; their trucks and box cars and airplanes, where they are, what they are carrying, how they are being maintained; and all of the materiel, supplies, and services required to keep large organizations operating smoothly.

Little Lost If Fail

The final reason for emphasizing the "obtain and use as is" option is simply that relatively little is lost if the attempt fails. If one completes the evaluation and selection process described below and fails to find a suitable DMS, he has a sound justification for considering the "obtain and modify" and the "build" options. More importantly, the knowledge gained regarding the strengths and weaknesses of existing systems will provide a sound technical basis on which to consider the other options.

Of course, the default option that a DMS is not required always exists. One may determine that normal source coding techniques are best, that a host-language system is needed, that the solution is beyond the state-of-the-art, or that one simply cannot afford what is desired.

RECOMMENDED APPROACH TO EVALUATION AND SELECTION

Two related dangers should be avoided when conducting a DMS evaluation and selection. The first danger is to avoid the explicit recognition of the importance of subjective judgments; the second danger is in not taking advantage of the iterative nature of the evaluation and selection problem structure.

Subjective Judgment Factor

The evaluation of software, particularly software as complex as many data management systems, is still an art not a science. Many judgments in an evaluation and selection of DMSs will be highly subjective. This fact is emphasized by explicitly considering the subjective judgment factor which will be invoked during the evaluation and selection process.

The subjective judgment factor is that weighting and appraisal of DMS features and performance which cannot be adequately described objectively. The fact that subjective judgments will be necessary should be anticipated at the start of an evaluation. This explicit, rather than the usual implicit, treatment of subjective judgments will assure that subjectivity is recognized as such by personnel conducting the evaluation and by personnel assessing the recommendations for final selection.

The subjective judgment factor will incorporate all of the knowledge, experience, and intuition of the individuals conducting the evaluation. It therefore seems prudent to assign the best personnel available within the organization to the task of evaluating and selecting a DMS. Depending upon the circumstances, it may be wise to supplement these people with the best assistance available from outside the organization.

The ultimate user of the DMS should be an active participant in the evaluation and selection process. The closer subjective judgments are made to the ultimate user of the DMS, the better he will realize the significance of these judgments for his particular situation, and the more likely that his interests will be fully understood and considered in making these judgments. This, in turn, will make more likely the ultimate success of DMS within the organization.

Iterative Process

A detailed checklist of procedures for evaluating and selecting data management systems is, in the author's opinion, not practical and perhaps not even feasible. The number of possible combinations of alternatives, options, and decision-making points is too large for enumeration. The nature of subjective judgments and decisions cannot be anticipated or described in the detail adequate for such an approach. In addition, a detailed procedural and decision checklist does not take advantage of the natural iterative structure of the evaluation and selection process.

The importance of an iterative evaluation and selection process cannot be overemphasized. A checklist approach tends to require a linear or sequential set of steps, i.e., step one followed by step two followed by step three. In some situations, however, correct decisions can be made based on little detailed data, while other situations may require substantial data for a similar decision. With an iterative process, one may iterate among steps, i.e., complete step one to a level of detail necessary to begin step two, then return to step one if greater detail from it is required for decisions during step two. By iterating between data collection and decision-making, one can minimize the effort for time consuming data collection for each evaluation because data is collected only to the level of detail required by the particular evaluation. A checklist approach, in general, must collect data to the level of detail necessary for the most complex evaluation likely to be encountered, and is difficult to tailor to particular less complex situations.

Framework for Evaluation and Selection Process

The evaluation and selection process described below takes advantage of the iterative nature of this process, and also allows the explicit consideration of subjective judgments. The process described should be considered as a framework, and guidelines for its use are presented by examples of the level of detail for data collection, and decisions which can be made based on such data.

Five steps form the framework for the evaluation and selection of a DMS. These should be conducted iteratively both among steps and within steps. The steps and their purposes are summarized, and then each is discussed separately in more detail.

1. Establish objectives for the use of data management within the installation or organization. This step outlines management goals for a DMS.

2. Identify the data management functions and features to be supported by a DMS. This step will begin to state, in terms independent of any particular DMS, the basic functions and features required to meet the objectives of Step 1.

3. Outline cost considerations. This step has two purposes. The first is to enumerate the types of dollar, manpower, and hardware resource costs which must be considered during acquisition and continued use of a DMS. Actual amounts for these costs will be established whenever they become known during the evaluation. The second purpose is to establish the amount of resources which can be afforded for the DMS.

4. Select/eliminate candidates which are likely/unlikely to fulfill the requirements and constraints established during Steps 1, 2, and 3. Specific data management systems, their features, and the functions they support are considered only to the level of detail necessary to identify a few good candidates for detailed evaluation in Step 5.

5. Conduct a detailed evaluation of the candidates remaining from Step 4, and make a final selection based on this evaluation.

The purpose of Steps 1, 2, and 3 is to develop the requirements and constraints for data management independent of any particular data management system. The purpose of Step 4 is to consider available data management systems in the light of the requirements and constraints established in previous steps, and to reduce the number of candidate DMSs to a reasonable number for a detailed evaluation. Step 5 will complete the evaluation and selection process. As discussed before, one should iterate among the above steps, particularly the first four steps, to perform the evaluation as efficiently as possible.

Step 1: Establish Objectives for Data Management

The first step in the evaluation and selection process is to establish the broad objectives and a simple statement of management goals for data managment within the organization. The range of possibilities is represented by the following three examples:

i. Continue operating essentially the same as at present but use a DMS to reduce manual effort and to provide a more responsive handling of requests for special reports.

ii. Consider changing current operations slightly. Provide on-line query for a small set of selected data which is otherwise available only through ad hoc conventional programming and with a response time of hours or days.

iii. Consider changing operations completely from a batch-oriented system to an on-line real-time interactive system.

The risks involved increase greatly as the range from examples i. to iii. is traversed. For any possibility selected, one should consider doing a complete "systems engineering look" at his problem independent of and prior to considering a DMS. A data base engineering effort, for example, would examine all files in the data base, the structure of the files, the processing and amount of activity against each file, the redundancy of data among files, interfile relationships, and the overall organization of the data base. This analysis could indicate that a restructuring of the data base and associated processing is desirable, and this, in turn, could decrease the need for a DMS or dramatically alter the features required of a DMS.

Typical examples of management goals are: decrease the average required skill level of an organization's programmers; provide on-line updating of critical files; reduce special report production cycle elapsed time; provide non-programmer management support personnel with convenient means of maintaining data and producing summary reports.

When planning to change the current operations significantly, one should also carefully consider alternatives for phasing into a new operation. A data management system, just as any other complex hardware or software system, requires checkout and shakedown for both the system and for the procedures for using the system. Personnel should be introduced gradually to a new system, and, in turn, a new system should gradually be introduced into the normal conduct of business. A "turn-key" phasing plan can lead to considerable initial confusion or even to a complete disruption of operations.

Step 2: Identify Data Management Functions and Features

Based on the objectives and goals established in Step 1, Step 2 will begin to identify the general functions to be performed by a DMS (for example, file generation, input editing, report generation) and specific constraints which a DMS should meet (for example, file structure, hardware constraints, mode of operation). Questions should be resolved regarding who will use the system (programmers, non-programmers, middle management, clerks); the files to be handled by the DMS (number, names, sizes, structures); the nature of the products to be produced (tabular reports, selective response to limited query); and the response time desired from request for product to product delivery (seconds, hours, days).

The above considerations should be made independently of any specific DMS, and should not be developed in great detail until absolutely necessary. A first pass at Step 2 will identify obvious

functions and features with little detail, and subsequent iterations among Steps 2, 3, and 4 will indicate the refinement of detail necessary for proper decision making. Early iterations should probably consider only essential functions and features. It may be reasonable to consider "would be nice" functions and features in later iterations, but "would be nice" features must not be allowed to become an unrealistic wish-list.

Step 3: Cost Considerations

Step 3 has two goals -- to anticipate the types of costs that must be considered for a DMS and to decide how much the user can afford for a DMS. Again, the level of detail will vary depending upon the number of iterations required by a particular evaluation and selection. The types of costs to be considered should be enumerated, but the amounts will be determined during other steps of the evaluation. How much can be afforded for each type of cost should be decided, and the amounts can be refined or changed based on findings during other steps of the evaluation. Exact amounts are often difficult to determine but all costs should at least receive a subjective consideration.

Costs for a DMS are of two types -- one-time and recurring. One-time costs include costs for selection, acquisition, installation, training, and one time data base conversions when necessary. Recurring costs for both hardware and manpower resources are often overlooked. Hardware resources include primary and secondary memory which must be dedicated to the DMS, at least when it is running. Additional computer time may be required for generating and maintaining files and for producing products with a DMS rather than with conventional production-oriented programming techniques. Considerable additional storage may be needed due to file expansion caused by some DMS file structures and physical storage strategies. Manpower resources will be required to provide readily available expertise on the use of the system and single-point responsibility for some system maintenance (such as reloading in the event of a system crash). Depending upon circumstances, in-house manpower resources may be required for DMS software maintenance and training of users on a continuing basis. Contractual support for continuing maintenance may also be necessary. Some cost tradeoffs or alternatives must be considered, such as lease versus purchase of a system.

An additional cost which must be considered includes the impact of a DMS on normal operating procedures for a computer facility and is extremely difficult to estimate. For example, supporting an on-line DMS may degrade batch operation turnaround significantly. Maintaining files through a DMS may increase problems of providing adequate backup or restart procedures after system failures.

Step 4: Candidate Selection

The purpose of the candidate selection step is to select a reasonable number of candidates for detailed evaluation by comparing the requirements and constraints from Steps 1, 2 and 3 with existing DMSs. This step may be conducted as a selection of candidates or as an elimination of unlikely candidates depending upon circumstances. It is at this step that an iterative approach is likely to save a substantial amount of effort. Often, many DMSs can be eliminated on the basis of a small amount of revealing data, and an iterative approach will take full advantage of this situation. For example, if all DMSs are eliminated which do not run on the hardware available, the list of candidates may be reduced quickly. If initial selection shows no satisfactory DMS, one may decide to relax slightly the requirements and constraints from previous steps in order to select candidates. For example, if one is seeking a DMS which will run on a 256K byte IBM 360/40 with two disc drives under DOS, he may be forced to consider adding more core, another disk drive or switching to OS in order that several DMSs qualify.

Candidate selection consists of an iteration among the previous Steps 1, 2 and 3 and among the following tasks:

a. Convert information describing DMS functions and features desired, and hardware, manpower, and dollar resource constraints, into a statement of constraints for a DMS which would fulfill all requirements. The level of detail of the statement of constraints will vary during iterations. It may begin as a single item, such as the hardware on which the selected DMS must run, and develop into a detailed list if many DMSs meet the constraints.

b. Identify all potential DMS candidates and consider each in terms of the constraints stated above.

c. Eliminate/select, by comparison with the constraints, candidates which clearly are not/are suitable. This elimination or selection should be only on the basis of constraints for which no compromise can be made (such as, the hardware on which the DMS must run, or the support of a file structure essential to the user.)

After the candidate selection is completed, one should have a reasonable number of DMSs (probably about three) for which to conduct a detailed evaluation.

The key to the success of the above process is the careful statement of constraints and the iteration among steps. Steps 1, 2, and 3 identify goals, state desires and constraints in general terms,

and outline major factors to be considered. Step 4 begins to develop details specific to particular DMSs. During candidate selection, high level general requirements, constraints, and features should be emphasized, with detailed requirements or features considered only if necessary to reduce the number of candidates to a reasonable size. Thus, for example, the machine mainframe and core requirements would be considered during candidate selection, but the number of levels of nesting of Boolean operators within the query language would probably not be considered until the detailed evaluation of Step 5. An example of the level of detail typical of an early iteration among steps is shown on Table III.

Step 5: Detailed Evaluation of Candidates

The final step in the evaluation and selection of a DMS is to conduct a detailed evaluation of candidates remaining after candidate selection. The detailed evaluation considers two factors -- the features of candidate DMSs, and the performance of candidate DMSs. By considering features, one determines if the DMS can do the job at hand; by considering performance, one determines how efficiently and effectively the DMS can do the job.

Several techniques have been used in the past for the detailed evaluation and selection of data managment systems. These usually consist of the development and analysis of a checklist of DMS features and some attempt to incorporate a performance factor.

The checklist analysis often weights very detailed DMS features numerically and performs some mathematical manipulations to obtain a score for a DMS. Unless the selection of weights and the manipulations are carefully and explicitly tempered by the subjective judgment factor, results have an air of precision, accuracy, and mathematical soundness which is not justified. In addition, features which have little or no importance for the particular application at hand, or features that are interdependent, tend to strongly influence the sensitivity of the results. Feature checklists can, however, be useful as reminders or guidelines for features to be considered during an evaluation.

The greatest single weakness of existing evaluation techniques has been the treatment of the performance factor, even though performance may be the key to the success or failure of a DMS during operational use. Performance factors have usually not been considered at all, or have been extrapolated from a combination of rules of thumb, values developed from models, or very limited benchmarks. The extrapolation attempts to predict how fast a DMS will perform various

TABLE III

EXAMPLE OF TYPICAL LEVEL OF DETAIL FOR
EARLY ITERATION OF EVALUATION PROCESS

Objective of DMS Use:	Reduce by 50% the man-time now expended for generating ad hoc reports from existing files.
Computer Resources:	IBM 360/40, 512 K bytes core, 3-2311 disk drives, OS/360.
	Willing to dedicate up to 30% of machine resources to the DMS.
Manpower Resources:	Willing to dedicate 1 system's programmer half-time to operating and maintaining the DMS, 3 programmers full-time to using the system to produce ad hoc reports.
Dollar Resources:	Prefer $20,000 to $40,000 purchase cost (or equivalent rental), including installation, training, and documentation.
	Vendor or other continuing support about $2,000 per year.
General Requirements:	The system will be used by programmers to produce one-time reports from existing formatted files that are also used for other production work. The system must therefore be able to access the existing files directly with no intermediate conversion or dual maintenance of files, and be able to produce reports similar to current ad hoc products. Reports must be produced within 24 hours after receipt of request, preferably in a batch environment.

functions for a particular application. This is an extremely difficult problem when the DMS must, in normal operations, compete for resources with other processing in a multiprogramming environment under the control of a complex operating system.

In addition to how fast a system performs certain functions, a performance factor should consider the human and operational aspects for a DMS, such as the ease of use of the system by the people who must implement the applications. This problem is not adequately addressed by existing performance techniques.

The approach recommended for detailed evaluation is simply to run real problems against the candidate DMSs and to select a DMS based on the experience gained. This is done as five basic tasks.

a. Select a set of file handling problems representative of the applications for which the DMS will be used operationally. In some cases, a single problem may be adequate.

A representative file handling problem consists of a body of data and typical products to be produced based on the data. The body of data may be an existing file, and typical products may be existing products generated from the file. The body of data must be large enough to provide realism, and should be the entire file in the case of existing files.

b. Arrange to run each candidate DMS. This may be through another organization that has the DMS, through a service bureau, or through a special trial period arrangement with the vendor. It is preferable to have the DMS in the facility at which the DMS would run if it were selected.

c. Run the representative file handling problems on the candidate DMSs. Exercise all functions and features identified in previous steps. Normally, this will include using the DMSs to generate the files, to manipulate the files for updating, maintenance, and retrieval, and to produce typical products from the files. Document the experiences encountered as the systems are run, including problems, man-time required for various activities, and hardware resources used.

d. Discuss the experiences for each candidate DMS with its vendor or other organization intimately familiar with both the external features and the internal implementation of the DMS. This will uncover gross errors or inefficiences which may have been made in using the DMS, and will validate the experiences and performance of the system for the problems run.

e. Evaluate the systems and select the system which was best for the representative problems. This evaluation will consider the features of the various systems, both claimed and experienced; the performance of the systems, both claimed and experienced; the time required to conduct all aspects of the evaluation; and, based on the experience, the subjective judgment factors for each system, such as ease of use.

Rationale for Approach

The approach recommended is not basically different from approaches used in the past, but the emphasis has been shifted substantially to provide a better understanding of the subjective judgment factor and of the performance of a DMS in a realistic environment. The approach described has several goals and advantages. First, it minimizes the amount of effort and time required to select a DMS from among, perhaps, dozens of candidates. Data collection is minimized by iteration with decision-making, and good judgment based on experience is accepted in place of detailed analysis. Second, the approach forces the people most critical to the success of the DMS, the people who will run and use the system, to become closely involved with DMSs. The user will learn more from this approach than any other about how to use a DMS, what it is good for, what it is not good for, and how it will affect him operationally. The user will also be in a better position from using this approach than from any other to understand and make the subjective judgments which are now inherent in software evaluation.

This approach is also likely to prevent many surprises which would otherwise be discovered only after acquisition. Some systems simply will not perform as might be expected. Some features of systems will not work as expected. Some features may not work at all.

The key advantage to this approach, however, is that it offers the best means of obtaining a good indication of a system's performance under realistic operating constraints and against real problems. Trying to estimate performance based on paper studies, rules of thumb, modeling or limited benchmarks is difficult. Even the people who should know a system best, the vendors, will usually require a substantial amount of data describing a problem before they will attempt to estimate how a system will perform on the problem. When given, the estimate may be couched in caveats and assumptions about the hardware environment while the system is running.

The cost of conducting an evaluation in the manner outlined above is difficult to estimate for a non-specific case. There is no substitute for experience during the iterative process of the evaluation.

A person with detailed knowledge of data management systems technology, and experience with many DMSs, will be able to conduct the candidate selection process very rapidly in most situations. The cost of gaining access to systems and for the necessary machine time to evaluate them may range from a few hundred to a few thousand dollars per system. Some vendors, for example, will contract for a one to three month trial period during which they will install the system, provide training, and perhaps assist in implementing a problem for around two thousand dollars. The cost of machine time may be "free" if the user's machine is used for the evaluation or could cost a few thousand dollars at commercial service bureau rates. The cost of manpower to conduct the evaluation will vary with the number of candidate DMSs, the number of representative problems, and the complexity of the problems and systems. To adequately evaluate a single problem on a single system could require anything from a few man-days to a few man-months, depending on complexity and completeness. In this area, it is important again to realize the difference between getting a DMS to produce a set of products from a given file and in evaluating the DMS against a representative problem. The former can require only a few days, especially with vendor assistance, while the latter requires carefully planned testing and documentation of most system features and their sensitivity to various parameters of the representative problem.

The cost for the evaluation and selection of a DMS will be a small percentage of the total expenditure for operating and maintaining the DMS during years of use. A careful and thorough evaluation will be a good investment.

Experiences

The Command and Management Systems Department of The MITRE Corporation is currently developing experience pertinent to the approach to evaluation described above. This work involves three existing data management systems and three problems selected from Air Force data processing facilities. A set of documents is being produced describing the systems, the problems, and the experiences of running the problems on the systems. The work was begun on an IBM 360/50 and is being completed on an IBM 370/155 at MITRE.

During this effort, each DMS has produced some unexpected problems. Bugs have been discovered. Some features do not work as expected. Some experiences are not yet fully understood. Several examples illustrate this experience.

One system was used to produce tabular formatted reports from a flat file of about 53,000 57-character records. During file generation, a sort of the file was necessary which was under control of

the Job Control Language (JCL) of Operation System 360. Extensive JCL was supplied as a part of the system, with parameters to be set by the user. Attempts to generate the file failed for six consecutive runs, apparently due to lack of work area for the sort. Before each run, work areas were expanded until the sort finally worked. In order to sort the file, a total sort work area of more than fifteen times the initial file size was allocated. The file, once generated, was more than three times its original size. The causes of this experience are being investigated.

Another system was being run against the same file mentioned above. The report generation language for the system proved to be so difficult to use that using COBOL would provide much faster response to requests for reports.

A third system had a feature for producing multiple reports in batch mode with only one pass of a file. Experience showed that this feature, which was supposed to improve performance in all cases over a single pass of the file for each report, in some situations performed considerably worse than passing the file once for each report. The vendor was contacted, he verified the results, checked his source code for the system, and decided some reworking of the code would correct the problem.

Several operational problems were encountered in running the DMSs. One DMS, when used for on-line queries, often degraded the overall computer throughput to an extent that the DMS was thrown off the machine by the operators in order to process their other workload. All systems encountered problems with time in primary memory partitions on the machine. Depending upon the nature of other jobs competing with the DMS for machine resources, the DMS would spend between a few minutes to tens of minutes in the partition to complete the same or a similar job. When partition time approached the latter extreme, again the operators would terminate the DMS run to make the partition available to other work. These problems, of course, are not the fault of the DMS, but are typical of what may happen when a DMS is introudced into an operation environment.

Elapsed time required to make the DMSs operational at the MITRE facility ranged from one week to two months. Time to run a problem against a DMS and document results ranges from two man-months to six man-months.

RECOMMENDATION

If everyone in the Air Force who needs a DMS goes through all of the steps described in this paper in order to evaluate and select a DMS, there will obviously be a lot of duplication of effort. This will be particularly true in identifying potential DMS candidates and in making the preliminary selection of candidates. Several paths can be pursued for eliminating some of this duplication of effort.

Catalogs and surveys of available data management systems can be compiled and made available to all prospective DMS users. This would eliminate the time-consuming and tedious task of developing a list of candidates each time a DMS is going to be selected. Such catalogs remain useful only if maintained on a continuing basis.

A team of highly experienced personnel could save potential DMS users a great deal of time and effort in the processes leading to the detailed evaluation of systems. The team could help the user document his DMS requirements, to organize and schedule the evaluation, to perform the candidate selection and to select representative problems for the detailed evaluation.

A convenient means of providing access to systems for detailed evaluation should be established. A similar means of discussing and validating detailed evaluation experiences with individual systems is also desirable. A variety of means are possible, ranging from contractual arrangements with vendors to the establishment of a government maintained inventory of systems and personnel experienced in their use.

It may be possible to establish a set of representative problems which would cover most of the prospective uses of DMS within the Air Force. Experience with these problems would be valuable for users during the evaluation, and would be especially useful to the team who assists users during the evaluation.

SUMMARY

Better techniques for the evaluation and selection of complex software should be pursued, especially for determining the performance of such systems. In addition, long-range planning for data handling within an organization should recognize that new developments in data handling software and hardware are likely to have a substantial impact upon the nature of traditional generalized data management systems. For example, many of the functions and features of these systems will be incorporated into host language systems or into operating systems.

This paper has described an approach to the evaluation and selection of data management systems for immediate application to existing data handling problems in the context of a single installation or organization. The approach is based on the rationale that a sufficient number of systems are available having a wide range of abilities and relatively low cost that one cannot afford not to consider using an available system.

The current state-of-the-art of evaluating and selecting complex software, such as data management systems, is heavily dependent upon the careful consideration of highly subjective factors. This fact is accepted and exploited by the approach discussed in this paper. The approach basically shifts the emphasis of previous approaches from paper studies and limited benchmarks to a great deal of hands-on experience with candidate systems by the prospective users of the DMS.

The use of existing data management systems, and the approach to evaluation and selection of these systems described in this paper, provide today's best opportunity for a user to obtain a satisfactory data management system at an acceptable cost.

REFERENCES

1. CODASYL Systems Committee Technical Report, Feature Analysis of Generalized Data Base Management Systems, May 1971

2. Survey of Program Packages - Report Generators and File Management Systems, Modern Data, August 1970

3. Auerbach Computer Software Reports, Auerbach Information, Inc.

4. G. J. Koehr, Preliminary Survey of Data Management, Reporting and Retrieval Systems, MITRE Working Paper (In publication).

The MITRE Corporation
Bedford, Massachusetts

AN EVALUATION OF THE GIM DATA MANAGEMENT SYSTEM THROUGH EXPERIENCE WITH MAC CARGO RECEIPTS DATA

Thomas R. Meier
Command and Management Systems

Sponsorship - Air Force Contract Number F19628-71-C-0002

INTRODUCTION

As part of the current Air Force Electronic Systems Division data management systems development program, The MITRE Corporation is running, on existing data management systems (DMSs), file-handling problems selected from operational Air Force environments. This activity has two goals. The first is to assess the suitability of various types of DMSs to handle various classes of file-handling problems. The second is to provide inputs for an approach to evaluating and selecting DMSs which satisfy a potential user's specific, immediate requirements for data management. This paper describes the experience of using the Generalized Information Management System (GIM) for a time-based, transaction-oriented scenario dealing with cargo-handling data of the Military Airlift Command (MAC). It also presents an overview of the process of selecting, preparing, and running a realistic problem under a particular DMS, and the experiences which may be expected from such a process.

DMS BACKGROUND

GIM was developed by the TRW Systems Group, Redondo Beach, California, and is currently being marketed in its IBM 360 and UNIVAC 1100 series implementations. GIM is a terminal-oriented DMS which maintains files on disk and provides both on-line and batch capabilities for input, updating, and retrieval of file dictionaries and data via the GIM Access Language. GIM is in the general class of DMS, referred to as self-contained systems, whose file generation typically requires conversion and internalization of user data, and in which any interaction with the data base is through the language provided by the DMS, rather than by independent user routines.

The work described in this paper was begun on an IBM 360/50 and was completed on the 370/155 at MITRE. The paper begins with a brief discussion of the original Air Force problem environment, followed by a description of the problem preparation process and scenario generation. Next, the steps required to prepare GIM to run the problem are outlined, and the scenario running and subsequent validation and evaluation activities are described. The paper concludes with a summary of the experience.

PROBLEM BACKGROUND

The Military Airlift Command (MAC), as an element of the Defense Transportation System, is required to maintain data bases and reporting systems in accordance with the Military Standard Transportation and Movement Procedures (MILSTAMP) and Military Supply and Transportation

Evaluation Procedures (MILSTEP). The Cargo Documentation Subsystem records the movement of each cargo shipment through the MAC airlift system.

Currently, the handling of cargo data within MAC is largely a manual operation. Cargo passing through a port is recorded by moving punched cards from box to box on the air terminal manager's desk. Card decks associated with cargo and flights are inserted into cargo containers and are forwarded to centralized reporting facilities after the requisite punching, duplicating, grouping, and stapling. Data for planning, scheduling, analysis, and reporting is retrieved at the centralized facilities by various COBOL programs run on RCA Spectra 70 equipment.

This manual processing produces high error rates; one source reports that typographical errors occur in the data for as many as 60 percent of the cargo shipment units. A historical file used in this work showed an error rate of approximately 10 percent for two types of detectable errors.[1]

To provide required information more efficiently, much of the current manual cargo data handling will be automated as part of the MAC Integrated Management System (MACIMS). The work described in this paper is based primarily on the description of the Cargo Documentation Subsystem in the tentative MACIMS specification, which documents current practices as well as their proposed upgrading.

The point of view taken here is centered on the activities of one port and the reporting of these activities to some centralized data processing facilities; this is essentially the air terminal manager's viewpoint. The areas of concern roughly parallel the chronology of cargo passing through the port. When cargo arrives, an entry is created in a cargo-on-hand file. Updates to this file allow status information to be provided for cargo at any point in the system. The load planner uses information in the cargo-on-hand file to schedule cargo departures by satisfying various constraints: priority; special handling; cargo size; weight, and compatibility; and allowable cabin load. This activity produces pre-manifests, which are lists of cargo shipment units to be loaded on each mission. When cargo is shipped from the port, the intransit control file provides the basis for monitoring mission execution. All subsequent monitoring takes the form of summary and evaluative reporting, and comprises the so-called "management transactions" on the cargo files.

---

[1] Uniqueness of transportation cargo number, and alphabetic characters in numeric fields.

PROBLEM SELECTION AND MODE OF OPERATION

The scope of the overall port cargo-handling activity just described is very broad, so it was necessary to select a subset of this activity to run under GIM. Choosing such a subset is perhaps the central issue involved in running a DMS with a realistic problem for evaluation purposes, since a primary goal of the evaluation is to demonstrate that the DMS can handle the range of processing required by a full implementation of the large or complex problem, but without accruing the cost of performing a full implementation.

The sources of the data and descriptions for the problem were as follows:

1. A historical data tape describing cargo shipment units passing through Travis AFB during a three-month period,[2] obtained through the MACIMS group at MITRE.

2. Descriptions of processing activities, assembled from current regulations and procedures in effect for aerial ports, such as MILSTAMP, and from the preliminary MACIMS specification.

3. Validation of various assumptions made in the course of the activity, as well as helpful suggestions, obtained from discussions with members of the MITRE MACIMS group.

The nature of the available data, primarily the history tape that described cargo entering and leaving the port, suggested the creation of a time-ordered sequence of events, or scenario, to form the basis for a model of the data-handling activities associated with cargo movement. Such a model would require as input a representation of the events that would have resulted in the history records found on the original data tape, a scenario whose use would model or imitate MAC practices. Many MAC activities, such as logging cargo into the port, posting manifests, querying port level data, and so forth, could be represented as on-line terminal inputs to a DMS. Accordingly, a preprocessor was built to generate an event stream combining these activities from a description of the desired scenario.

---

[2] 66,780 cargo shipment units, representing 4166 flights; all but 253 cargo shipment units date from days 32 to 120 of 1968.

The following "primary" events were generated directly from the MAC cargo receipts history tape data:

1. Cargo arrivals.

2. Pre-manifest posting (the primary activity of the load planner).

3. Flight departures.

For these events, a standard ten-hour elapsed time between pre-manifesting and flight departure was selected. The subsetting of the original data records to produce the primary events is illustrated in Figure 1.

The following "secondary" events were generated from sources describing typical cargo-handling activities at an aerial port.

1. Aperiodic (ad hoc) inquiries: frequently concerned with status information or with port level data, these information requests occur on an "as required" basis.

2. Periodic inquiries: these retrieval requests recur at regular intervals for such scheduled events as for daily briefings on special-handling requirements of the port, or weekly inventories.

3. Subsets and histories: these are occasionally extracted from the files for further processing (perhaps beyond the DMS's manipulation capabilities) or for forwarding to other facilities. One example would be the set of inactive records which was extracted and then purged from GIM. This subset is analogous to the original cargo receipts data from MAC.

DMS ACTIVITIES WITH THE PROBLEM

The steps required to prepare GIM to receive the scenario inputs are:

1. Analysis of the data and definition of the items and files required.

2. Preparation of the dictionary inputs and of the disk areas which will contain the files.

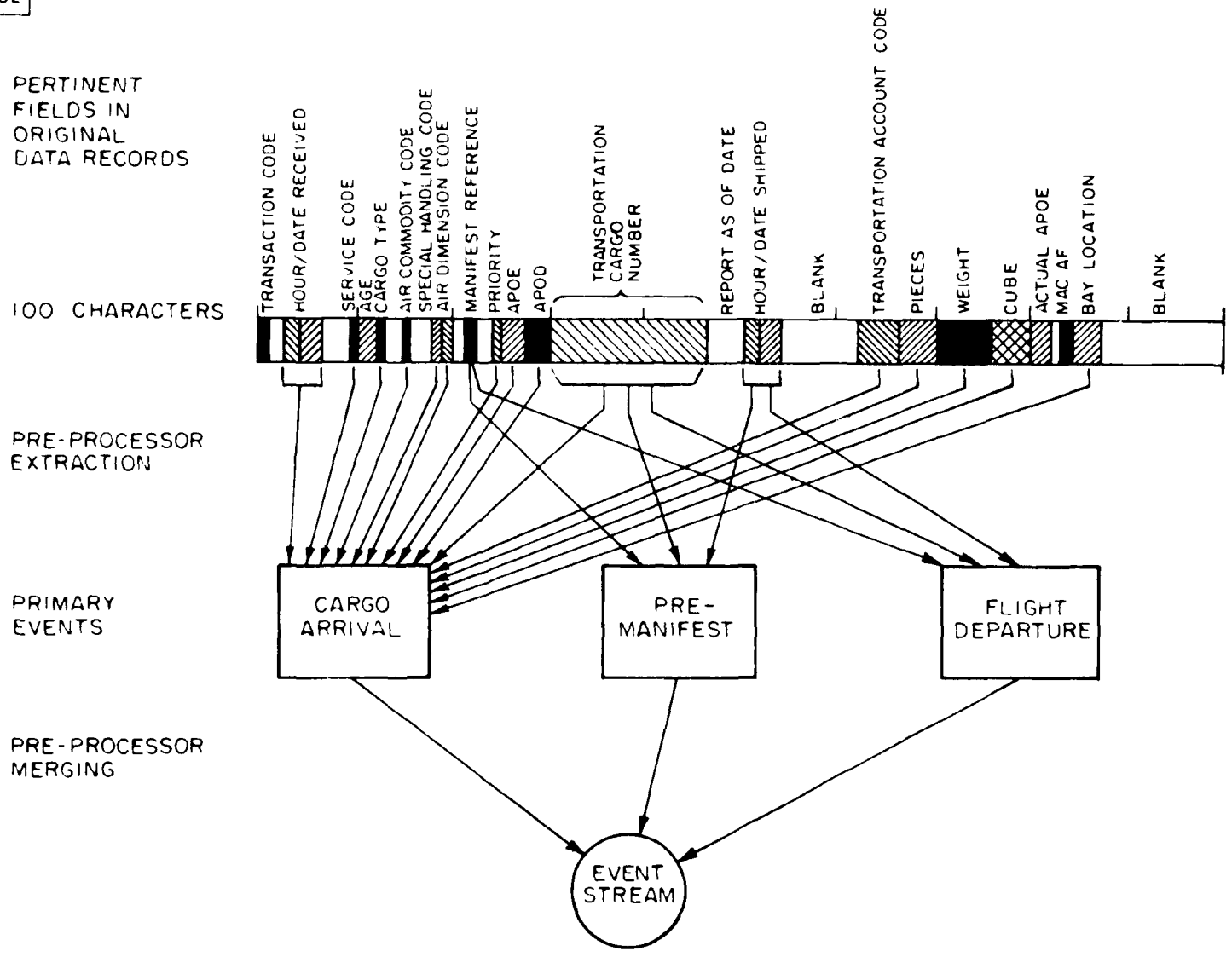


Figure 1 CONCEPTUAL VIEW OF PRIMARY EVENT INTER-RELATION

3. Loading the dictionaries and data via the GIM Access Language.

The terminology used in describing file generation has been made as general as possible, with the express interest of avoiding the unusual TRW nomenclature. Thus, a file will comprise many records or entries; records or entries will be identified (named) by primary items or keys; and dictionaries define the formats and usage of fields or items within an entry. For instance, the transportation cargo number (TCN) will be used as the primary item of the cargo-on-hand file, and the values for all items, such as pieces, weight, or time received, that pertain to a given cargo shipment unit will, together with that unit's TCN, comprise an entry in the cargo-on-hand file.

Each step required to prepare the model, and to prepare GIM to run the model, will be discussed in terms of the activities involved, the problems incurred, and the resources required.

SCENARIO GENERATION

Scenario generation will be described first. Figure 2 shows the origin of each segment of the event stream that will be input to GIM. The scenario generation program creates primary events from sorted copies of the original MAC data: cargo arrival notices from a tape ordered by time received at the port, and lift notices and pre-manifests from tapes in time-shipped order. These last two input tapes are identical, but it should be recalled that ten modelled hours elapse between their uses of the same original record. Secondary events are described to the scenario generation program by card input which contains the following information:

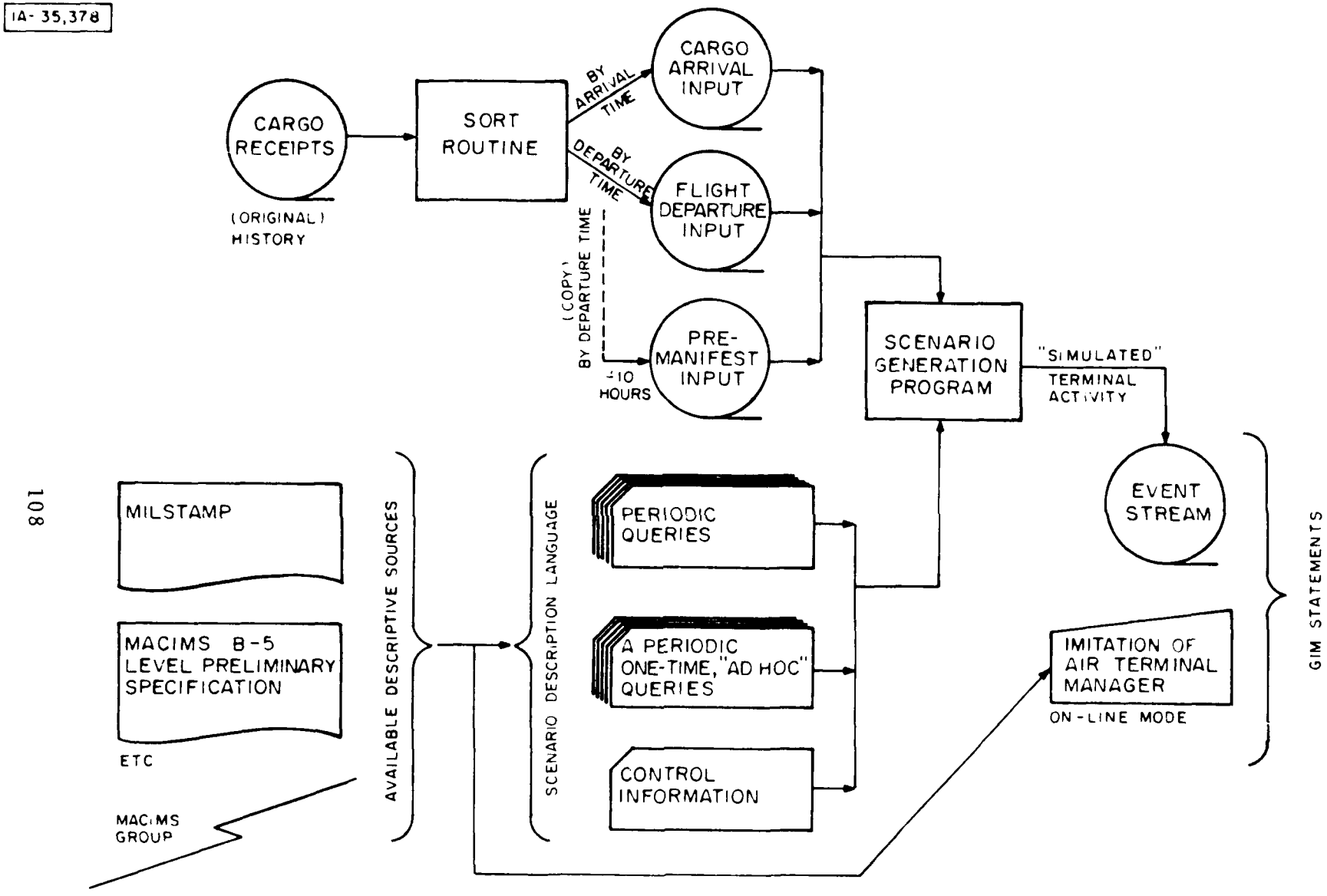
1. Whether an event is periodic or aperiodic (one-time).

2. If periodic, at what interval it should occur; if aperiodic, at what time.

3. The fixed content of the input statement(s).

4. The method of calculating any variable content of the input statement(s) (optional).

Examples of all event types are shown in Figure 3.



801

Figure 2 PROCEDURAL VIEW OF PRIMARY AND SECONDARY EVENT INTER-RELATION FOR THE SCENARIO GENERATION PROCESS

Cargo Arrival

```
ADD TCN 'AT81VE00246009XXX' TIME-REC "33*10"
SERVICE "A" CTYPE "3" AIR-COM "V" SPC-HDL "Z" AIR-DIM
"D" PRTY "R" APOE "SUU" APOD "DAD" TR-ACCT "A205"
PCS "1" WGT "765" CUBE "4" TAR "802" #
```

Pre-Manifest

```
ADD MANIF 'DX' ETD "33*19" LINE
"AT80MP00100009XXX" "AT860200061122XXX"
"AT87RW00100017XXX" "A5820200160001XXX" #
```

Flight Departure

```
ADD TO MANIF 'CX' ATD "33*9" TDP "801" #
```

Periodic (for example, every day at 0800 hours)

```
TOTAL THE AWGT PCS AND CUBE FOR ALL TCN WITH
NO DAY-SHP#
```

Aperiodic (for example, on day 33 at 1600 hours)

```
COUNT ANY TCN WITH AIR-COM EQ "P" ANDD WITH NO DAY-SHP #
```

Subset for Further Processing and Purge

```
EXTRACT ANY TCN WITH DAY-SHP LESS/THAN OR EQ TO "31" #
DELETE ANY TCN WITH DAY-SHP LT = "31" #
```

Figure 3. Examples of Event Types

The scenario generation program extracts appropriate fields from the primary event input tapes, and inserts the queries as indicated by the scenario description input decks, completing calculations based on the simulated time where required. Control information defines such things as the days to be modelled and whether port activity will have to be initiated prior to the model period (i.e., filling the port with cargo). The last source of events for the scenario, also from the descriptive sources, is the extended imitation of an air terminal manager's activities, entered from an on-line terminal, to which one may switch at any convenient "time" in a scenario run, as shown in Figure 4.

DATA ANALYSIS AND DICTIONARY PREPARATION

The first step in preparing GIM to run the scenarios, starting from the problem description, involved the analysis of data requirements. The items required to maintain the implemented capabilities fall into four categories:

1. Cargo descriptors (physical characteristics of the shipment unit).

2. Static processing descriptors (which remain constant for a given shipment throughout its processing).

3. Chronological processing descriptors (which accrue to each shipment unit as it passes through the system).

4. Other descriptors specific to a given function, such as utilization data, mission data, or intransit correction codes.

The availability of each of these items is shown by category in Figure 5. The model was restricted to those activities which could be supported by the data available. For example, mission status could not be maintained beyond the knowledge that a certain pre-manifest had or had not been issued, or that the flight associated with a pre-manifest had or had not departed the port.

The next step was to determine the interrelationships and formats of the data items, and to decide what files to create and how to group the data items into files. The usual method for specifying such relationships for GIM is a data base map. The data base map for the cargo files is shown in Figure 6, with arrows indicating inter-file and inter-item retrieval and updating.

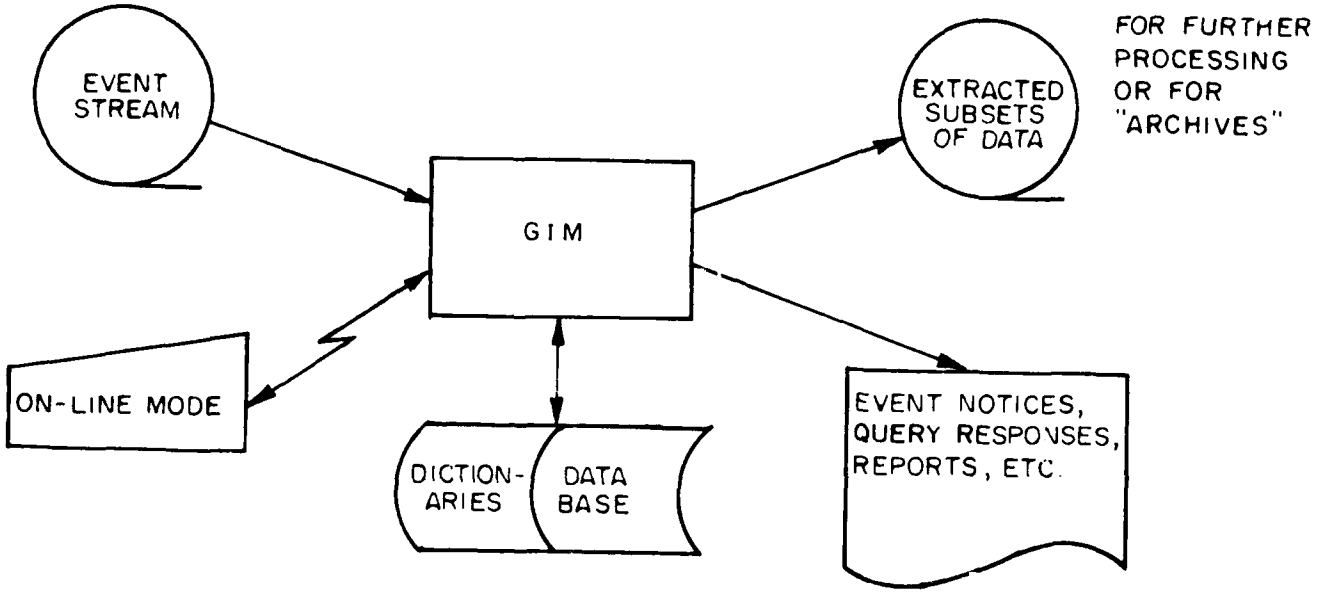


Figure 4 RUNNING GIM WITH SCENARIOS

Category	Item	Available On Cargo Receipts Tape?	Explanation
Cargo description			
	transportation cargo number	Y	unique cargo shipment identifier
	number of pieces	Y	
	weight	Y	tons (to 4 decimal places)
	cube	Y	volume in cubic feet
	type of mail	N	
	air commodity code	Y	type of cargo as K for parachutes, V for vehicles, etc.
	air dimension code	Y	smallest craft loadable
	special handling code	Y	as Z for none, W for refrigerate, etc.
Static processing description			
	APOE	Y[1]	aerial port of embarkation
	APOD	Y	aerial port of debarkation
	mode of transportation code	N[2]	
	service branch	Y	F for Air Force, N for Navy, etc.
	transportation account code	Y	account paying for movement of materi l
	project code	N	
	consignor	N	
	consignee	N	
	required delivery date	N	
	priority	Y[3]	1, 2, 9, or R
Chronological processing description			
	hour/date shipped	Y	
	hour/date received	Y	
	hour/date processed	N	
	bay location	Y[4]	
	pallet information	N	
	shipment designators	N	
	hour/date departed	Y	
	hour/date released	N	
	manifest reference	Y[5]	
	carrier	N	
	age	Y[6]	

Figure 5. Data Item Availability

Category	Item	Available On Cargo Receipts Tape?	Explanation
Other			
	utilization date	N[5]	
	mission data	N[5]	
	correction codes	N	

Notes

[1] always SUU = Travis.

[2] known to be F = MAC.

[3] full range of values not given; 3 and 4 grouped with 2.

[4] not used locally (Travis).

[5] although no mission data is known, the cross-referencing function of the manifest reference item is used to group shipment units together to form their implied, modelled manifests.

[6] not used, however, because an internal GIM function was available to calculate the age in port at departure time.

Figure 5. Data Item Availability (Concluded)

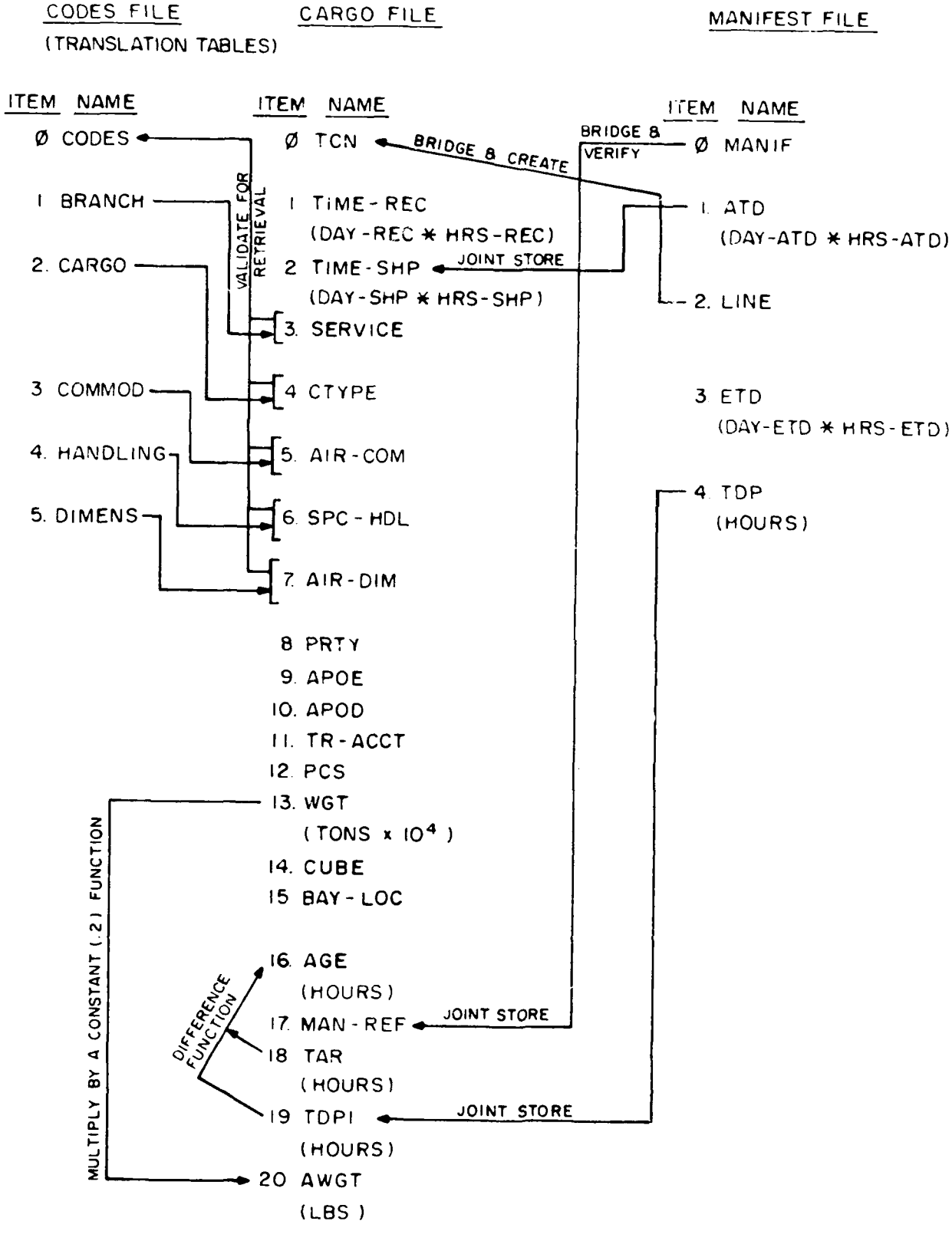


Figure 6 GIM DATA BASE MAPS

The preparation of the data base required an analyst to select and define the problem activities and to develop the data necessary to support them. The analyst must be familiar with the candidate DMS and must interact with operational personnel whose points of view he will need to know in order to describe, refine, and validate the problem definition. In general, this step may be reviewed whenever re-analysis is indicated by the exigencies of actual runs with the candidate DMS.

In fact, this step was reviewed several times in preparing for the scenario, because problems with inconsistent data and data formats could not all be handled by the scenario generation program. For example, the occurrence of non-unique cargo shipment identifiers was not anticipated in the preliminary file design. Allowing a cargo shipment to have multiple values for number of pieces, weight, and volume fields was considered the best quick solution, causing the least disturbance to the port level data. In an operational environment, of course, a non-unique cargo identifier would cause GIM to ask the terminal operator to check his typing.

When data maps have been prepared, a programmer must then generate the procedures and inputs to structure the files. For GIM applications, this involves coding at least one GIM Access Language statement per item or field in each file. These statements add entries in a GIM user dictionary, which is itself a file, defined by the master dictionary. Thus the language used to create dictionaries is the same as that used to load data into GIM. The relation between dictionary and data files is illustrated in Figure 7. All of the transformations of the dictionary data, from analyst to key-punch operator/typist are illustrated in Figure 8.

DISK PREPARATION

The next step prepares disk areas for the data base and creates the master dictionary. GIM files are disk-resident, with entries packed into data structures called GIM physical records. This step allocates available disk areas and divides them into segments. These segments, the GIM physical records, are the storage units addressable by the GIM software routines.

When the disk areas have been formatted and the master dictionary has been created by GIM, each file is initialized and a master dictionary entry is added for it. Next, user dictionaries are entered, and finally, system synonyms are created. Typical entries in the cargo dictionaries are shown in Figure 9.

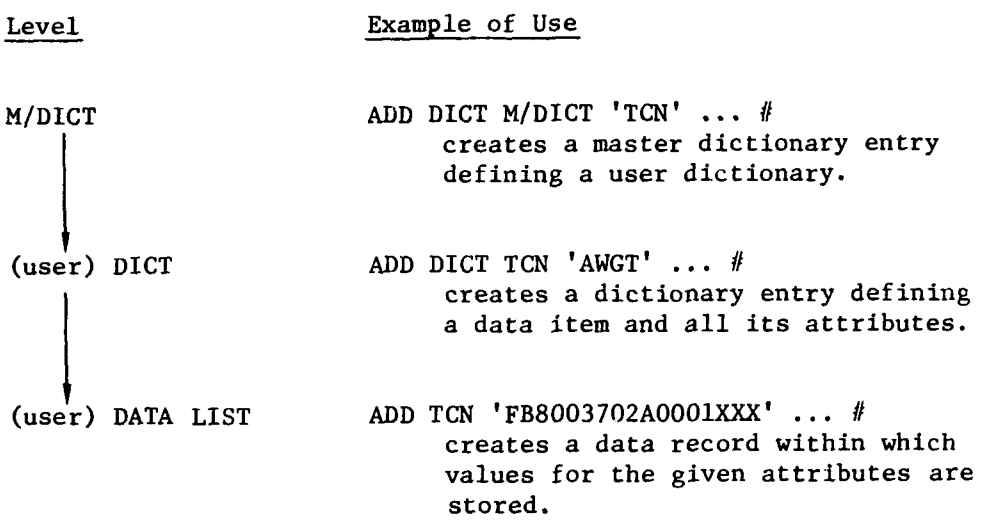
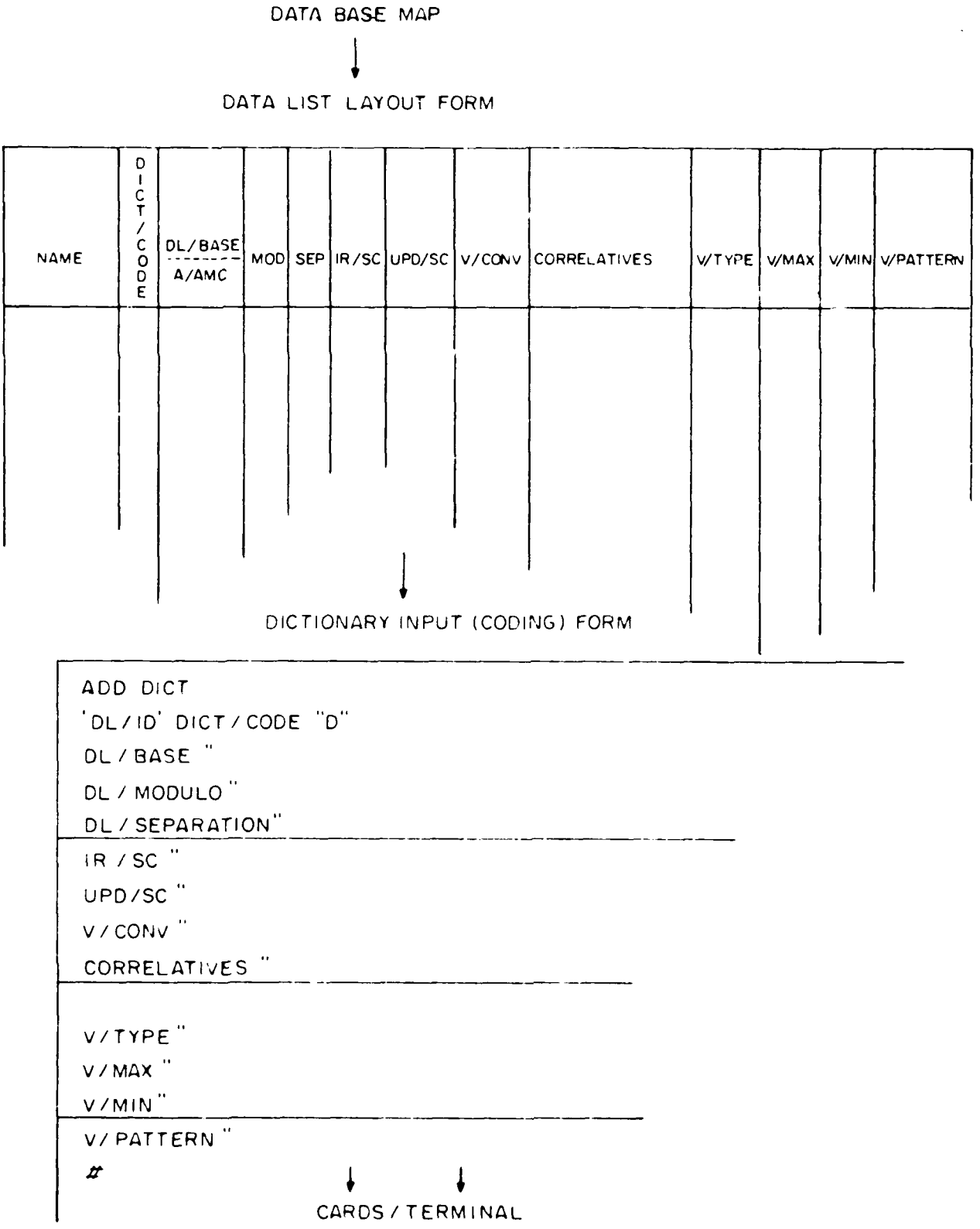


Figure 7. File Levels in GIM



IA-35,379

Figure 8 DICTIONARY PREPARATION PHASES

User Dictionary

Data values in this file describe the values that may be stored in the file of user data.

	Notes
TCN : 10 DICT/CODE : S DL/BASE : 10 DL/MODULO : APOD DL/SEPARATION : 10 DL/CORRELATIVES : Y211 DL/TYPE : LA DL/MAX : 3 DL/MIN : 3	Defines a system synonym for the tenth item of dictionary TCN, APOD. Single-value, replacement by new values, left-justified, alphabetic, of length 3.
TCN : PCS DICT/CODE : A DL/BASE : 12 DL/CORRELATIVES : Y112 DL/TYPE : RN DL/MAX : 3	Defines the twelfth item of dictionary TCN, PCS. Multiple values allowed, right-justified numeric field of maximum length 3 characters.

User Data

TCN : AT80TS00096058XXX TIME-REC : 32*22 SERVICE : ARMY CTYPE : GENERAL AIR-COM : CONSTRUCTION&BLDG MATERIALS SPC-HDL : SUBFREEZG.REFRIG. AIR-DIM : UNASSIGNED PRTY : R APOE : SUU APOD : PHJ TR-ACCT : A205 PCS : 1 WGT : 70 CUBE : 3 AGE : -790. TAR : 790 AWGT : 14.00	Lists all values for the cargo shipment unit defined by this TCN. AGE is defined as time shipped less time received, which will be illogical for TCNs (such as this one) which are still in port.

Figure 9. Dictionaries Compared to Data

DATA LOADING

When dictionaries have been completed, data may be loaded. To initialize the data base for scenario runs requires entering data for all the cargo that would be in port at the beginning of the first modelled day. A control variable input to the scenario generation program allows the option of creating in the event stream the GIM statements to add such cargo, so that running the first part of the scenario will "load" the port.

Several problems were encountered with the data loading process. First, estimating the disk area requirements and finding the best organization for these areas is difficult, because the method of storing any data into GIM files involves a hashing algorithm over which the user has little control. The number of data "buckets" (hash points) and the size of the "buckets" (number of GIM physical records per hash point) are controlled, but finding values for these parameters which are suitable to the data is a trial-and-error process.

Second, data loading had to be repeated several times, whenever structural changes to the files had been made. The character-by-character storage of item values in GIM caused the value " 35" to be unequal to "35", for example, and a dictionary modification was made to reject on input values with the former format. Before this change was made, some surprising query responses were received.

Another problem area associated with initial loading of user data was the attempted use of the GIM system verb BULK, whose acceptance of data inputs from a tape permits considerable savings in GIM overhead over statement-by-statement loading. The "standard GIM tape format" expected is unusual, and several other problems, including lack of explicit documentation, compounded the difficulties experienced with this feature. Several telephone calls to TRW programmers, and discussions with other GIM users, failed to resolve all the details of BULK use.

Relatively little human effort was required to load the data, except for BULK tape production. Data input costs experienced in an operational environment would be due to keypunching or typing at a terminal and would be linearly related to the volume of data. Machine time is also reasonably linear, until the disk area organization begins to cause many overflow records, a situation which can be remedied, by finding better hashing algorithm parameters, as described earlier.

Computer resources required for port initialization to a typical level (over 1500 TCNs) are estimated as follows: approximately two seconds of 360/50 processor time per TCN, increasing to about two-and-a-half seconds as the file fills.[3] (The 370/155 processor is estimated to run three to four times faster than the 360/50.)

On the average, thirty-five disk accesses were required to add a cargo shipment unit to the file, but this figure would vary widely depending upon the amount of memory available to the GIM system.[4] Also, frequent checkpoint dumps were taken, requiring as many as 800 additional disk accesses and 45 seconds of 360/50 processor time. Restoration to checkpoints was frequently required because operating system crashes or input/output device problems would often interrupt updating procedures, leaving inconsistent data in GIM files and rendering GIM inoperable. These rough timing estimates apply to the statement-by-statement addition of file entries, and hence are also assumed to apply to file updating during model runs.

SCENARIO OPERATION

The scenario runs combine time-ordered file updates from the input event stream with data retrieval requests, which may also be entered from the input event tape or from the on-line terminal. Figure 4 shows the relationship of the various inputs and outputs involved. The sequence of occurrences initiated by each of the primary event types is as follows:

1. Cargo arrival: usually a TCN file entry is created for the arriving cargo shipment unit, and all available cargo and static processing descriptors are stored. Time received, the only chronological descriptor applicable to cargo arriving, is also stored. For cargo shipment units previously manifested, these descriptors are added to the existing TCN file entry.

---

[3] At the highest port level, about 20 per cent of the cargo involved overflow records (entries that would not fit in the "bucket" assigned to them by the hashing algorithm). A better organization of the disk areas later reduced the average ADD time to below two seconds.

[4] Batch jobs at MITRE normally receive a 128K core partition; although GIM can be forced to run with significantly less, degraded performance should be expected.

2. Pre-manifesting posting: an entry is created in the manifest file, which lists each TCN to be shipped on this flight and the estimated time of departure. The manifest reference is inserted into each TCN entry affected, with TCN entries being created for cargo which is not yet in port.

3. Flight departure: the appropriate manifest file entry is updated with the actual time of departure (ATD), which is also entered into each TCN file entry associated with this flight. The ATD is stored both in day/hour shipped format and as a number of hours (for the age computation).

Secondary events, of course, involve those data items specified in the retrieval requests. Figure 10 shows a typical sequence of primary and secondary events as found on the model input tape, and illustrates system responses.

Every modelled day produced an eight- or ten-inch high printout, representing the combined actions of the cargo handlers logging cargo, the load planner posting mission schedules, the production of lift notices, and any transactions of the air terminal manager. History tapes were extracted, or dummy tape units were specified to the model to suppress the actual writing of these tapes. Evaluation of scenario runs, wading through this printout and mapping the extracted data tapes, turned up several problems, some of which involved returning to preparation steps because of changed assumptions about the available data or about the GIM documentation.

Most problems encountered in this stage, however, were of a more operational nature: For example, the manner in which GIM software is implemented for OS/360 occasionally hampered operations. Whenever a tape was to be required during a run, even if only for taking a checkpoint after several hours of on-line activity, that tape had to be mounted for the duration of the job, physically tying up a tape drive. More careful macro-instruction issuing would have allowed the user to defer this mounting, a standard option in the OS/360 Job Control Language.

The resources required to run the model for one day at a large cargo terminal like Travis can be summarized as follows: Six cylinders of 2314 disk storage (about 875,000 characters) were allocated for files, but less than two-thirds of that space was formatted into GIM physical records. At highest port level, GIM used 9 per cent of the available GIM physical records (about 52,000 characters); the cargo files (including translation tables) occupied 70 per cent (about 448,000 characters); and 13 per cent (about 75,000 characters) was still available. The density of data in the occupied records

```
TRANSACTION 2687
STATEMENT ADD TCN 'FB526000230601XXX' TIME-REC "33*9" SERVICE "F" CTYPE "3"
AIR-COM "V" SPC-HDL "Z" AIR-DIM "H" PRTY "1" APOE "SUU" APOD "HIK"
TR-ACCT "F8AO"
PCS "6"
WGT "360"
CUBE "6"
TAR "801"
#
@FB526000230601XXX@ ADDED.

TRANSACTION 2688
STATEMENT ADD TCN 'AT87FX93386502XXX' TIME-REC "33*9" SERVICE "A" CTYPE "3"
AIR-COM "V" SPC-HDL "Z" AIR-DIM "S" PRTY "R" APOE "SUU" APOD "SGN"
TR-ACCT "A205"
PCS "1"
WGT "20010"
CUBE "135"
TAR "801"
#
@AT87FX93386502XXX@ ADDED.

.
.
.

TRANSACTION 2704
STATEMENT ADD MANIF 'DP' ETD "33*19" LINE
"BVSCAP00191843XXX"
"DVSHBP00280001AXX"
"FB526993144781XXX" ...
#
@DP@ ADDED.

TRANSACTION 2705
STATEMENT ADD MANIF 'DQ' ETD "33*19" LINE
"AT80MP92814003XXX"
#
@DQ@ ADDED.

.
.
.

TRANSACTION 2716
STATEMENT ADD TO MANIF 'CX' ATD "33*9" TDP "801" #
@CX@ UPDATED.
```

Figure 10. Event Stream Inputs and Responses

```
TRANSACTION 2717
STATEMENT ADD TO MANIF 'CY' ATD "33*9" TDP "801" #
@CY@ UPDATED.
:

TRANSACTION 2722
STATEMENT LIST ONLY EACH TCN AIR-DIM WGT TIME-SHP
WITH DAY-SHP EQ "32" #

TCN.............  TIME-SHP  AIR-DIM  WGT...

FB44480030X138XXX   32   3   C-124         5
N004079357H001XXX   32   3   C-124      6400
R0462993573350XXX   32   3   C-124        75
:                   :    :   :            :

FB528000052569CXX   32  22   UNASSIGNED  295   (two cargo shipments have
                                         305    arrived with this identifier!)
FB526400290127XXX   32  22   C-119       880
:                   :    :   :           :

****

TRANSACTION 2724
STATEMENT COUNT ANY TCN WITH NO DAY-SHP ANDD WITH SPC-HDL EQ "W"#
NUMBER OF ACCEPTABLE ITEMS = 32
TRANSACTION 2725
STATEMENT ADD TCN 'FB528100283196XXX' TIME-REC "33*10" SERVICE "F" CTYPE "3"
AIR-COM "A" SPC-HDL "Z" AIR-DIM "V" PRTY "2" APOE "SUU" APOD "VCR"
TR-ACCT "F8AO"
PCS "1"
WGT "900"
CUBE "14"
TAR "802"
#
@FB528100283196XXX@ ADDED.

TRANSACTION 2726
STATEMENT ADD TCN 'FB525000263205XXX' TIME-REC "33*10" ...
:
```

Figure 10. Event Stream Inputs and Responses (Concluded)

was between 70 and 80 per cent, which produced much of an observed 52 per cent expansion of file size from the original history tape. The remaining expansion is attributed to system overhead and built-in redundancy, such as the numeric fields which contain a number of hours duplicating hour/date items, but which were required for age-in-port calculations with the limited arithmetic functions available in GIM. This density falls within the range recommended for efficient updating.

As might be expected, more computer resources were required to run the scenarios than for any other stage of the exercise. Over 65 per cent of the processor time and 87 percent of the total disk accesses used were utilized in running fewer than 200 modelled hours and in "loading" the port four times. In general, of course, these amounts depend greatly on the number of times structural changes to the files were made and reloading was necessitated, and on the number of different scenarios actually run.

The on-line mode of operation was a useful adjunct to the event stream input from tape. Response times were best when qualifications involved primary items -- that is, TCNs or manifests queried by name; three to four second response time (measured between end-of-message transmission and the terminal commencing to type the response) was usual, with two-and-a-half seconds being the observed minimum (the time to return a syntax error message, for example). With queries involving simple selections based on qualifications of data values, other than for the primary items discussed above, twenty to forty second response times were not uncommon when searching the entire cargo file (at a typical load of 1600 TCNs). Twenty seconds was the observed upper bound for most queries selecting values from the manifest file (with 200 or more flights).

Problems involved with the on-line mode of scenario running were due primarily to its impact on the normal operation of the computer facility. The elapsed time during which GIM remained active on the computer (i.e., in a partition) was often so long that the computer operators would request that operation of GIM be terminated in order to free machine resources for other users. The lack of the deferred tape-mounting option (mentioned earlier) was part of this general problem.

The scenario-running stage required the time of computer operators primarily, and of programmers to arrange/supervise the activity and perform further processing on extracted data. The on-line mode required some effort as well, but mostly in the scenario-preparation (script-writing) step.

VALIDATION OF ACTIVITIES

Once some scenarios have been run, the next step is to begin to evaluate the results. This evaluation has two facets: first, determining the accuracy of the model, that is, how well it represents real activities. Second, verifying that the model encompasses the full range of processing required for real activities, including processing which, though not explicitly modelled, would be handled by a full implementation of the system.

Evaluating model accuracy is a counterpart to the problem selection process, in that it requires an awareness of the operational user's point of view to insure that the modelled products correspond in essential details to the actual products. There is an element of feedback, of course, whereby the experience with the DMS, or merely the planning of DMS usage, will suggest alternative practices, especially in procedural areas. For example, some routine report generation might be obviated by the availability of an on-line querying capability. This could completely change the specific requirements to be satisfied by the DMS. Such possibilities should be anticipated, and the requirements to be met by a DMS should be open to modification during the entire conduct of the evaluation.

Another example of potential benefit based on the experience with the cargo data is the impact of GIM's input validation feature. Even with the few checks made, it is possible to envision how DMS usage in MAC's Cargo Documentation Subsystem could affect error-tracing procedures. Catching typographical errors while logging cargo into the port, for example, would imply considerable reduction in the volume of error-tracing messages sent over the communication lines.

A major goal of running a realistic problem is to exploit the available data management functions as fully as possible within the scope of the evaluation process. This is accomplished by direct contact with the vendor, with other users, and with personnel involved in the current activity for which a DMS is being planned. All of these sources of information were utilized in running cargo receipts scenarios under GIM. The MITRE MACIMS group rendered assistance during the problem preparation and verification stages; TRW was called upon to explain details lacking in the user documentation, and those parameters or procedures that were installation-dependent; and TRW staff and other MITRE users of GIM (not involved with the cargo receipts activity) were influential in improving the GIM implementation of the application.

The approach taken for this problem toward report generation under GIM, for example, was the direct result of the advice of another GIM user. Producing a GIM report involves extracting a file subset and running it interpretively against another GIM file which describes the processing and formatting of the report. The other MITRE GIM user estimated that the ratio of computer time used, between this mode of operation and running a COBOL program against the extracted data subset, is on the order of hundreds-to-one. Also, the preparation of the GIM files which describe the processing and formatting of reports entails the use of an extremely complex description language. These factors dictated the emphasis toward on-line aspects of the cargo receipts problem and the assumption that external processing would be done as required, especially for standard report production. The effort required for the problem would probably have been magnified significantly by any attempt to include the Generalized Reporting Capability of GIM.

SUMMARY OF ACCOMPLISHMENTS AND COSTS

At some point, the process of refining the correspondences between the modelled and the actual activities must cease, so that what could and could not be done in support of the problem by the DMS may be summarized. The GIM model is thought to represent a fair sampling of every type of processing that might be required of a data management capability for MAC cargo-handling. This claim must be tempered by a statement of the constraints on this problem. The data available and the distance from the users of the actual, current system whose capabilities were modelled placed some atypical limitations on this activity as an evaluation of a candidate DMS. (The goals of this activity, it should be remembered, did not include the recommendation of a DMS for MAC's Cargo Documentation Subsystem.)

GIM showed several strengths. Those aspects of cargo-handling facilitated by availability of immediate, current information were handled well by the model. The load planner's and the air terminal manager's interactions with the files recommend themselves to this mode of operation. Ad hoc queries (of certain types) would also be more convenient to run. And operational upgrading, such as logging cargo arrivals via an on-line terminal, seems to be a straightforward implication of GIM use. The volume of transactions did not appear excessive for the GIM system, either; a typical day of over fifty flights and about a thousand cargo shipment units could be run in just over two hours' elapsed time on the average.[5]

[5] On the 360/50, under OS with MFT; scenarios were normally run during the night, but with moderate-to-heavy system load.

The weaknesses of GIM for the cargo receipts activity can be summarized in two areas: the limitations of the access language (and its documentation) and the relatively restricted interface available to further processing (because of the specialized tape format). For example, many desired queries contain phrases like "by channel", "by priority", or "by cargo type". The sorting of selected records that this implies is not available in the access language, although it is part of the Generalized Reporting Capability.

The documentation provided to MITRE by TRW contains no explicit definitions of the GIM Access Language syntax. Exposition proceeds by examples of typical uses of most verbs, giving little guidance as to their restrictions and subtleties. For example, the hierarchy between logical operations in qualification clauses cannot be guessed from the examples and assurances given in the user documentation. Only the simplest cases are covered. The absence of a centralized source (document or person) of information about GIM implementation under OS was also annoying. But, it should be remembered that these are only procedural difficulties encountered in "learning the system". GIM supplied the basic capability to support all those activities which were definable within the constraints of our activity, with a basic weakness only in the access to mass amounts of data. Perhaps our reluctance to force GIM to create all the standard cargo movement reports, which was a factor in this problem, would also be a consideration in a full implementation of a DMS for cargo-handling. Standard report generation might be accomplished by the continued use of current COBOL routines, with perhaps a preprocessor added to convert GIM-extracted data to the input formats required.

Overall, the activities described in this paper required approximately five man-months of effort and over twenty-two hours of 360/50 processor time.[6] A breakdown of the resource use by activity step is given in Figure 11. Compared to a full implementation the costs are small, and there is a high probability that, in addition to evaluating a specific DMS, DMS requirements will have been clarified, and the candidate DMS will have been a useful design tool.

---

[6] Estimation of processor utilization is hampered by the changeover from a 360/50 to a 370/155 configuration during the activity, but a ratio of 10 to 3 has been suggested, and is incorporated in the calculations given here.

Activity	Resources: Man-Weeks (rough estimates, due to overlapping of activities)	Computer: Runs	Processor (minutes)	Disk accesses (thousands)	Other
1. Data acquisition and analysis, problem selection	3 1/2	--	--	--	--
2. Data base preparation (disk)	1 1/2	15	17	45	fair amount of card input: M/DICT, etc.
3. Dictionary definition preparation and loading	2	21	22	29	small amount of card input.
4. Scenario generation, including design and implementation of scenario generation program	3	50	290	23	tapes generated, small amount of card input.
5. Scenario running	3	82 on-line: 12	878 56	1,266 25	much tape input.
6. Further problem analysis and structural changes	1	(minimal: grouped together with step 3. above.)			
7. Maintenance and error recovery activities	2	42	50	66	many checkpoint tapes generated and used.
8. Measurement and evaluation	2	48	24	10	minimal.
9. Documentation	4	--	--	--	
TOTALS	22 (weeks) (=5 man-months)		1,337 (minutes) (=22 1/4 hours)	1,454 (thousands)	

Figure 11. Resource Utilization Summary

The steps involved in preparing and running a problem activity against a DMS to provide inputs to an evaluation/selection process have been described for one of the standard problems and one of the representative data management systems being used at MITRE. The problems, achievements, and costs of the activity have been identified to provide the reader with some data for assessing this approach to evaluating and selecting DMS capabilities.

DESIGN/EVALUATION DISCUSSION GROUPS

During this symposium, three periods were set aside for discussion. The first two periods were concerned with the problems of designing and evaluating generalized data base management systems. Several systems were discussed in the light of personal experiences of some of the attendees. Nothing was solved, but the discussions promoted communication and pointed out the need for future symposia related to these topics.

The final day was a wrap-up session chaired by our distinguished discussion moderators, Dr. A. G. Dale of the University of Texas at Austin and Dr. Edgar H. Sibley of the University of Michigan. They related the topics of the symposium to the following subject areas:

Axioms for System Development

1. Information systems are capital investments.
2. Build success criteria measurement into an operational system.
3. The system is for the user.
4. Establish landmarks.
5. Don't be afraid to cancel projects.
6. Get management to participate where it counts.
7. System development is an iterative process.
8. Consider as many design alternatives as are feasible.

9. The great leap forward is best accomplished in short comfortable hops.
10. When in doubt...document.
11. Use documentation to structure the design process.
12. Establish checks and balances.
13. If you can't plan it, you can't do it.
14. Procedures are as important as programs.
15. Conversion is a system. (Cost of conversion is part of capital investment.)

Today's Research and Development

1. Development of storage structure languages.
2. Analysis of accessing techniques.
3. Extension of host language systems by self-contained functions.
4. Data base restructuring algorithms.
5. Data gathering on system performance and data base.

Problems of Large System and File Design

1. How is worth of information determined?
2. How is the user charged for storing his information, retrieving it and maintaining it?
3. How do we check the validity of data in the file?
4. How to determine the net worth of the system concerning 1-3.

5. How do we specify user requirements?
6. Given a set of data requirements, in what ways may data be structured?
7. Can a few general access methods be designed to interface all large and complex files efficiently?
8. Is the overload of GDBMS too great a range of user files?
9. Is is possible to define a language which describes the mapping of data to physical structure?

CONCLUSION

At this last session, it was the consensus of opinion of the attendees that there is a need for a technical working group to be formed. The purpose of this working group would be to provide a forum for discussions between personnel involved at the working level of research, design and implementation of generalized data base management systems. In addition, this group would publish its findings to provide management with a tool to aid in making decisions about these systems.

It was decided by the attendees that the USAF Academy should take the initiative in forming the working group and that information about the group should be available by the end of January 1972.

The symposium was extremely successful in providing a means for communicating between personnel involved in the design and evaluation of data base management systems. As a result, future symposia will be held at the USAF Academy in the area of data base management systems.

USAF ACADEMY WORLDWIDE DATA MANAGEMENT SYMPOSIUM

9-11 December 1971

LIST OF ATTENDEES

Air Force (ACD)

Cpt James J. Davern
AF (ACDB)
WASH DC 20330
221-0670/0660

Cpt Rickell D. Knoll
AF (ACDB)
WASH DC 20330
221-0670

Maj Jack Gill
AF (ACDC)
WASH DC 20330

Air Force Accounting and Finance Center

CMsgt Bennie Bauman
AFAFC (AD)
3800 York Street
Denver CO 80205
555-6644

Joseph D. Ryan
AFAFC (AD)
3800 York Street
Denver CO 80205
555-6845

AF Data Services Center

Cpt William Burrows
AF Data Services Center
WASH DC 20330
225-9147/9

Cpt Michael Parmentier
AF Data Services Center
WASH DC 20330
225-9147/9

Air Force Data Systems Design Center

Cpt Cecil Martin
AFDSDC (SYA)
Gunter AFB AL 36114

Lt Larry Ritchard
AFDSDC (SYA)
Gunter AFB AL 36114

Maj Robert Yorks
AFDSDC (SYA)
Gunter AFB AL 36114

Air Force Intelligence

Lt Col Donald H. Gregory
AF (INAT)
WASH DC 20330
22-75040

James H. Hoge
AF (INYMB)
WASH DC 20330
224-8485/8251

Air Force Logistics Command

Cpt James R. Chappell
AFLC/ACTAC
Wright-Patterson AFB OH 45433
787-4044

Gary F. Eckenrode
AFLC/ACTAC
Wright-Patterson AFB OH 45433
787-4044

Air Force Military Personnel Center

Maj Harry M. Kepner
AFMPC (DPMDDA)
Randolph AFB TX 78148
487-4266

Maj Thomas H. Lee
AFMPC (DMPDDA)
Randolph AFB TX 78148
487-4266

Air Force Systems Command

Cpt David C. Peterson
AFSC (ACDO)
Andrews AFB
WASH DC 20331
858-6407

Air Force Office of Scientific Research

Cpt Barry L. Haas
AFSC (AFOSR)
1400 Wilson Blvd
Arlington VA 22209
224-5101

Air Force Command & Control

Cpt Raoul F. W. Roy
HQ USAF (AFXOOCSB)
Pentagon WASH DC 22030
227-7652

Air Training Command

Cpt Ralph B. Fritzsch
TSCOD/59
Sheppard AFB TX 76311
736-2917

Cpt Gary F. Hoff
TSCD/39
Sheppard AFB TX 76311
736-2917

Auditor General

Cpt Albert F. Allen
Auditor General
ADP RSCH & TNG Office
Box B3758, PSC 2
Sheppard AFB TX 76311
851-2636

Defense Intelligence Agency

Cdr Thomas M. Dykers
DIA (DS-5C2)
WASH DC 20301
22-25266

Electronic Systems Division

Lt Col Denis A. Conrady
ESD (MCDT) Stop 36
L.G. Hanscom Field
Bedford MA
478-5391 (5394)

MACV

Maj Patrick L. Harris
MACV (DMA)
APO SF 96222

Military Airlift Command

Maj Kenneth Pattison
MAC/ADSD
Scott AFB IL 62225
638-3136

MITRE Corporation

Barry L. Gerken
Mitre Corporation
Bedford MA 01730

John Jeffries
Mitre Corporation
McLean VA 22101

Thomas R. Meier
Mitre Corporation
Bedford MA 01730

Jonathan A. Singer
Mitre Corporation
Bedford MA 01730

Naval Postgraduate School

LCdr Robert M. Hanna
NPS (Code 53HQ)
Monterey CA 93940
479-2655

NORAD

Cpt J.O. Bernier
NORAD (XPCP)

Maj A.E. Biery
NORAD (XPCPA)
692-6145

NORAD (Con't)

1Lt Brian W. Cobb
NORAD (NPCP)
692-6454

E.E. Griffee
NORAD (NPCPA)
692-2571

R.A. Harris
NORAD (NPCPA)
692-2724

Cpt R.R. Heckman
NORAD (NPCP)
692-2708

G.A. Hemmings
NORAD (NPCPA)
692-2031

Maj B. Legette
NORAD (NPCPA)
692-2031

Ltjg Bardwell C. Moss
NORAD (NPPG)
Ent AFB CO
692-6454

Cpt Nicholas B. Powell
NORAD (NPCP)
692-6454

W.E. Smith
NORAD (NPCPA)

Rodney R. Stover
NORAD (NPCP)
692-6454

Cpt Joseph J. Thigpen
NORAD (NPCP)
692-6454

CMSgt Edward W. White
NORAD (NICA-P)
635-8911-2259

Rome Air Development Center

Joseph Cavano
RADC (ISIM)
587-3827

Ray Liuzzi
RADC (ISIM)
587-3827

Denis Maynard
RADC (ISFA)
Griffiss AFB NY 13442
947-3270

Frank Sliwa
RADC (ISIM)
587-3827

SAC

Maj Robert Carpenter
SAC (DOCD)
Offutt AFB NE 68113

Cpt William Janofsky
SAC (DOCD)
Offutt AFB NE 68113
271-4723

2Lt Jeffrey Krauser
SAC (DOCD)
Offutt AFB NE 68113
271-4714

1Lt Robert K. Miller
SAC (DOCD)
Offutt AFB NE 68113
271-4726

TAC

Robert S. O'Neale
TAC (DOCA)
Langley AFB VA 23365
432-3659

Cpt Stanley C. Valdez
TAC (DOCA)
Langley AFB VA 23365
432-3659

USAF Academy

Gary Maruska
USAF Academy (ACD)
259-3100

Joseph C. Rock
USAF Academy (ADC)
259-3360

USAF Academy (DFAST)

Col John P. Wittry

Lt C Monti D. Callero

Maj Gordon M. Gerson

Maj Clifford J. Trimble

Maj John A. Zingg

Cpt Bruce A. Burns

Cpt Michael P. Gyauch

Cpt Jack A. Mauger

Cpt James H. Nolen

Cpt Donald E. Willis

Cpt Anthony J. Winkler

University of Michigan

Assoc Prof E.H. Sibley
University of Michigan

University of Texas at Austin

Assoc Prof Alfred G. Dale
Univ of Texas at Austin

U.S. Naval Academy

Lt R.F. Belanger
Comp Sci Dept
USNA
Annapolis MD 21402
851-3450/367

Lt J.J. Draper
Comp Sci Dept
USNA
Annapolis MD 21402
851-3456/367