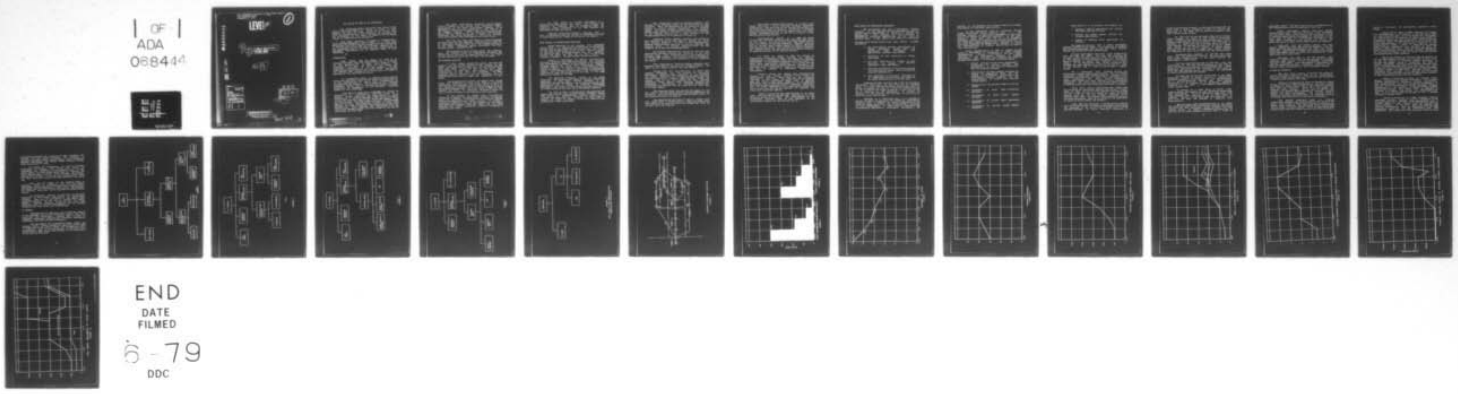
AD-A068 444 NAVAL INTELLIGENCE PROCESSING SYSTEMS SUPPORT ACTIVIT--ETC F/G 9/2
THE IMPACT OF DBMS ON ADP MANAGEMENT.(U)
MAR 79

UNCLASSIFIED NL

1 OF 1
ADA
068444



END
DATE
FILMED
6-79
DDC

NAVAL INTELLIGENCE PROCESSING SYSTEMS SUPPORT ACTIVITY
2461 EISENHOWER AVENUE
ALEXANDRIA, VA 22331

LEVEL II

(1) B.S

AD A068444

DDC FILE COPY

(6) THE IMPACT OF DBMS ON ADP MANAGEMENT.

(11) March 1979

(12) 27p.

410 508 LB

THE IMPACT OF DBMS ON ADP MANAGEMENT

New technology always institutes some change in the world of those who employ it. Sooner or later, it also changes the world of those who do not use it too. Many times the effectiveness of new technology is dependent on the willingness of its users to respond with the needed changes in organization and procedures.

This paper describes the impact of implementing a major database management system (DBMS) in a batch data processing environment and the changes which have transpired in the four years since IDMS was installed. Emphasis is on the management view of the change and what future trends in management requirements can be expected based on past experience.

INTRODUCTION

Even though they may appear the same on an organization chart, most ADP organizations are as different as the personalities of their staff members. This difference is, in fact, the result of the imprint and individual philosophy of the ADP manager. Few ADP organizations really achieve the "ideal" model. The reasons for missing that plateau are so varied that an attempt to describe or list them would be futile.

It is possible, though, to identify certain factors which have sufficient impact on the effective functioning of an ADP organizations to insure their consideration by conscientious ADP managers. The introduction of a database management system into an installation requires a careful evaluation, by management, of the potential long-term impacts of this technology.

Five years of experience in FMS/DBMS use places an organization at a particularly vulnerable place for a manager. Enough experience has been gained to begin establishing a trend of response to DBMS but not enough to clarify the long-term impacts upon future ADP support. This is particularly true when using a flexible, full-networked DBMS such as IDMS. The experience of users of other DBMS packages with less flexibility only generally parallels our observations. Their problems were associated with the limitations imposed by DBMS; ours are created by the lack of limitations in using IDMS.

The Naval Intelligence Processing System Support Activity (NIPSSA) is the primary data processing support organization of the Naval Intelligence Command. It is basically a service bureau, tasked with a wide variety of ADP support projects. NIPSSA came into existence in the early 1960's, operating an IBM 7090/1401 installation. Everything was batch processed in those days. Multiple files proliferated, many overlapping into other files.

The IBM 360 came into NIPSSA as a general purpose processor when the 7090 departed. Starting with a model 30, we progressed through a model 40 to the 1-megabyte model 50 we use today. Operating systems progressed from DOS, through OS/MFT, to OS/MVT V21.8F.

The description which follows traces the progress, pain, and pitfalls which we experienced following the fateful day in 1973 when it became evident that the tunnel we were following had no end, no light, and no future.

THE DATABASE DECISION

In 1973, NIPSSA maintained nearly 100 major ADP systems. All were batch-oriented, written largely in ANS COBOL. Some were over 10 years old. All had been converted from 7090 COBOL. Sixty per cent of the files were interrelated in some way. User requests for support involving two or more files was increasing. The most pressing problem, however, was the increasing cost of maintaining files which no longer were responsive to user needs.

A government-owned package, the Modular Data System (MODS) was installed in July 1973. MODS is a two-level hierarchical file management system (FMS) which, as long as the application is simple, does a fine job supporting small scale batch applications. The first lesson of DBMS use was learned with MODS: be selective. We weren't and we are still living with some of the disasters which resulted.

It was evident, by January 1974, that MODS was inadequate to support the large-scale database integration which was necessary. Investigation into available FMS/DMS software packages painted a bleak picture. We needed a networking capability such as that proposed by the CODASYL Data Definition Language Committee (DDLC) and the known packages were either too big for a 360/40 or incapable of supporting our needs.

At this point, an in-house development of a CODASYL-based DBMS package was begun. The software was beginning to function when we discovered IDMS. An exhaustive formal evaluation of all DBMS packages was performed and IDMS won easily.

IDMS was installed at NIPSSA in February 1975. A year of exhaustive work had resulted in procurement of what we hoped would be our "light at the end of the tunnel."

DBMS EFFECTS ON ORGANIZATION STRUCTURE

Dennis Appleton, in his January, 1977 DATAMATION article said: "... some major philosophical and managerial changes must be made before we can be sure that that light at the end of the tunnel is not a train." Within 3 months of installing IDMS, it became evident that Mr. Appleton, writing two years too late for us, had voiced a very prophetic statement. We were already riding the train.

Figure 1 illustrates the general organizational structure of NIPSSA when IDMS was installed. The structure was ill-prepared to support the development of an Integrated database spanning the entire Naval Intelligence Command. Database support was buried at the lowest possible level within the Command. In addition, the manager of application support (even though he participated in the evaluation and selection of IDMS) had placed all of his eggs in the MODS basket and refused to use IDMS.

The political situation proved to be mixed blessings. The systems staff began to develop standardized approaches to using IDMS. Without pressure to get production systems implemented, the systems programming team experimented with a number of software development approaches. The results of that effort are being felt today in a more effective and less costly development approach.

By June of 1975, a reorganization had separated the general ADP services from the specialized SEAWATCH system. (See Figure 2.) Unfortunately, database support still retained its lowly position. Progress in using IDMS to broaden the level of support to the user community was, predictably, almost nonexistent.

Such things have a habit of haunting managers. The pressure to get something useful for the money spent on IDMS increased. This was heightened by the failure of the MODS package to fulfill several requirements for database support. When upper management became aware of the problem, changes began to occur. The need to increase the authority of the database section was a major catalyst in the reorganization which is illustrated in Figure 3.

Comparing levels between Figures 2 and 3 does not show impressive gains for DBMS support. However, the gains were actually dramatic. As part of the reorganization, database administration came into its own as a separate and distinct entity. It received responsibilities and THE AUTHORITY TO GO WITH THE RESPONSIBILITIES.

The Database Administrator was now on a par with the applications software branches and reporting to the same division head. This reporting point is very important to successful DBMS progress. With only one level of management to go through, decisions and coordination is more effective and timely.

This organization remained largely unchanged until September 1978 when the structure was changed slightly. The impact of that change, however, was also dramatic.

Through 1976, 1977, and early 1978, we found that database management was the ultimate in frustrating experiences. Largely because a majority of the personnel and support resources for database development came from external contracting, database administration became the classic case of the tail wagging the dog. External influences caused impacts which seriously affected DBA performance. Centralizing all database coordination, scheduling, and analysis into one branch put too many eggs in a single basket.

The pressures became too much for the people in the DBA branch. The resulting turnover further complicated the procedure. The train had gathered speed gradually until it was becoming difficult to hang on.

The organization described in Figure 4 should have been implemented 9 months earlier. That hind sight might have spared us the loss of several good people and some of the pain which has resulted.

This latest organizational change is another step toward achieving a truly effective database managEment organization. The work load of the DBA has been broken out and distributed among the branches. Instead of all database activities being centralized at the branch level, the entire Systems Management Division is now, for all intents, a Data Administration organization.

Organizational impacts of a DBMS are often significant! Politics nearly buried a promising DBMS effort at the outset. And the politics were those INSIDE the ADP shop, not outside in the user community. External politics can be vicious when database begins to affect traditional organization structures. But, unless the internal conflicts are resolved first, you may never have the dubious pleasure of exposing yourself to the external battles.

It would be unfair to drop the subject of organization structures without defining the "better" organizational arrangement which is "ideal" for database development. If ten database administrators were asked for their ideal organization structure, it is likely that we would get at least eight different answers. Therefore, the structure which I illustrate in Figure 5 is only one opinion and must be adapted to individual organizational requirements.

Figure 5 places the overall data administration function at the highest possible level within the organization structure. While this is not practical initially in many cases, the interaction of the database upon the entire organization makes such a position desirable, and in some cases, mandatory. It is impossible for the data administration staff to interface effectively with the major departments of an organization unless they are on a par with them and can exercise, if required, the clout such a position implies.

Figure 5 places the Data Administrator on an equal level with the head of the ADP shop. This may be the first political battle to be fought. The effectiveness of the illustrated structure will be very dependent on the relationship between the ADP head and the DA.

DBMS EFFECT ON DEVELOPMENT MANAGEMENT

ADP development has been traditionally project oriented. As new applications were requested by users, a project team was established or assigned to develop the applications. This approach has been effective in implementing individual applications for the same or multiple users. It has, however, been subject to some major failings which become evident only upon careful, in-depth evaluation:

1. The relationships between applications is seldom considered except where the application under development must specifically interface with another.

2. Uniformity of data definition is poorly controlled.

3. The file definition is limited to that necessary to support the specific application under development.

4. Long-term maintenance is very costly because file modification causes order-of-magnitude resource expenditures.

5. Each application is typically developed as an independent entity without the advantage of common modularization of source code.

Database has been touted as the "light at the end of the tunnel." That light, on the front of our database train, will never help ADP management effectively unless we modify our approach to using it. If we turn around and look in the direction the light is shining, it will show us the way to improving our ADP operations. If, however, we continue to look directly into the light, it will blind us while the train runs us down.

Database is a functional tool. It is intended to provide the means to centralize the storage of information. Properly used, a database is largely independent of the input and output functions. This can be termed a variation of data independence. In this context, departing from long-standing application development philosophy, the

contents of the database are no longer based on the output requirements of individual applications.

In the summer/1978 issue of "Database", the ACM SIGMOD publication, Dennis Appleton says: "... the problems inherent in applications management are what gave rise to the concept of data base management. In the final analysis, 'data base' was, and is, a DP management philosophy intended to counteract the problems generated by its alter-ego, computer applications. Essentially, it is a functional management philosophy which attempts to ignore the individualities of computer applications in favor of their similarities."

Since implementation of IDMS at NIPSSA, we have worked on development of approaches to common database implementation. Figure 6 illustrates the most recent iteration of the development approach to database development. This PERT-type chart supports ALL levels of database development. It is broken into seven phases:

1. Initial analysis of the user's requirement, preparation of service analyses, and determination of the level of interaction with the existing database.

2. Definition of the data elements necessary to support the application which are not already in the database. Definition of the relationship between elements and the identification of logical data entities (record types).

3. Integration of the defined elements into the schema.

4. Development of batch input processing software.

5. Development of batch output support software.

6. Development of on-line input processing software.

7. Development of on-line output support software.

Using Figure 6, it is possible, for example, to:

1. Develop a complete application from scratch for batch, on-line, or both.

2. Develop new output support software for existing applications.

3. Expand an existing batch application to on-line.

The important point is that a common development approach applies in all cases. It is practical to schedule and estimate development time and cost with great accuracy because the process has been standardized.

Software development costs have risen dramatically over the past few years. For most ADP managers, the increase in the cost-per-instruction rate is not as apparent as the backlog in user requests for service. Quoting the costs of software is useful when fighting for a bigger budget. It is meaningless when trying to explain to a customer why their support is so long in arriving or, after it has arrived, why it takes so long to make a "simple" change.

As a functional tool, database permits the definition of common modularized software. This can produce significant results when properly applied. Figures 7 and 8 illustrate the reductions in labor hour requirements for input processing and output report development, respectively, which have been attained using modularized techniques. The charts depict the average effort required to develop individual input processing and report programs.

First let's look at input processing. Before IDMS was installed all application software was written in either COBOL or MODS. One of the first apparent failings of MODS was that it took as long to create an input processing program, with all the data elements validated, as was required under COBOL. The average input processing program, for batch processing, required an average of 160 labor-hours of effort to write, debug, and document.

When IDMS was installed, it became quickly apparent that the ultimate scope of database development would exceed our resources if we did not find a better way of writing database software. An initial experiment converting an

existing application directly to IDMS was terminated when we found that a direct conversion failed to utilize the advantages of the DBMS and, for practical purposes, made IDMS a faster access method.

By late 1975, we had developed COBOL source library books to support all types of data element validation. Common processing code which was used repetitively in every database input program was also "canned." The result was a reduction in effort from 160 to 70 labor hours to write an input program. The savings resulted from the use of common code, stored in the source library, to reduce coding, punching, and logic errors. A common input format reduced the complexity of input instructions and user documentation.

Continuing this approach, by mid-1977 the labor effort had been reduced to 24 hours. The improvement came from further modularization of the input function and improvements in IDMS itself.

The real break-through came in 1978 when, with the availability of the IDD data dictionary, we were able to begin utilizing the dictionary as a development aid. Input processing modularization was advanced to a new level in which the input program itself, using COBOL, was generated by another parameter-driven program.

We found that 80 per cent of the code in an input processing program is either constant or consistently formatted according to identical rules. The implementation of a program-generator reduces most input program definitions to an average of 10 parameters, depending primarily on the number of data elements supported on a single punched card.

An average of 4 labor hours to define, code, compile, debug, and document an input processing program may seem unbelievable. But it is true and it does work. Some input programs have required 20 or 30 labor hours to complete. Several, on the other hand, have been done in an hour with the most time consumed in preparing user documentation.

The next step, under development now, is to reduce the parameter preparation function by utilizing information stored in the data dictionary during analysis and data element design. We expect to completely generate over 90 per cent of all input programs with less than 2 hours of

programmer effort. The major hang up will be preparation of user documentation and we're working on that too.

Output development efforts have been reduced dramatically as well. The ease with which IDMS interfaces with the COBOL report writer facility was the primary factor in reducing output report development by one-third in the first year of IDMS use. Introduction of CULPRIT in 1977 made report preparation much quicker and easier and permitted users to begin writing their own ad hoc requests.

Since 1977, over 200 CULPRIT reports have been written supporting both tape files and database. Only one COBOL report has been implemented in the same time frame. Continuing improvements in CULPRIT have reduced the average report development time, including user documentation where applicable, to 6 labor hours.

What does this mean to the ADP manager? More effective support of our users can be provided in a shorter time span that before. In an increasingly constrained resource environment, such improvements in performance can make the difference between a successful operation and a failure (often accompanied by a loss in personal income).

A LOOK AT STATISTICS

The next seven figures illustrate the history of certain statistical impacts of DBMS on our organization. They show trends which, while not infallible, can be applied to many service-oriented ADP shops.

Figure 9 illustrates the level of personnel on board during the period of DBMS implementation. The total number of people is, today, exactly half of the resources available when IDMS was installed. Since 1977, the number has remained fairly constant. For our level of support, this appears to be adequate when augmented by external contract support for major developments. The in-house staff is primarily concerned with maintenance of installed systems and design and coordination of new system development.

The average experience level of in-house professional employees has cycled somewhat over the past few years. This is due, in part, from a policy of hiring at the bottom where possible. The experience trend, from January 1975 to today, is mildly upward. Hiring of experienced IDMS professionals from the outside has not been very successful.

Figure 10 illustrates the professional experience level trends.

A major dollar impact of DBMS, especially if coupled with teleprocessing, is an increase in training costs. DBMS is a very intricate area. To do the design and development job effectively, it is necessary to expose staff members to as broad an educational base as possible. Seminars and user conferences where the opportunity to talk with people from other organizations, are particularly important to the professional's growth. Figure 11 shows that the average annual training expenditure has increased by three and one-half times since 1975. Given the continued emphasis on DBMS, teleprocessing, and distributed database approaches, it would not be unusual to see the annual training cost for some professionals reach $2500.00.

One of the reasons we installed IDMS was because, of all the DBMS products available on the market at that time, it did not require a full-time systems programmer to maintain the system. Figure 12 illustrates the level of systems programmer interface which has been required since 1975. The most important point to be made from this chart is that the level of effort required by a systems programmer increases quickly as additional software products are interfaced with the DBMS. It is important that the vendor effectively coordinate new releases of DBMS-related products so that interfacing efforts can be minimized.

Figure 13 identifies the level of in-house effort devoted exclusively to database development. This includes database administration functions, contract coordination, user consultations, and data dictionary maintenance. It does not include maintenance of already developed database applications (1 1/2 persons) or systems software support (previously defined). This effort level is expected to increase as existing applications are redeveloped under IDMS. It will likely return to the peak of 4 to 5 persons achieved in January 1978.

Figure 14 rounds out the DBMS personnel resource commitment picture. External contract assistance is the only practical way to develop large applications with a minimum impact on the internal staff and still meet the user's deadlines. To be sure, there are hazards with external contracting. This is especially true with IDMS programming since not too many contractors have IDMS experience. This problem has been partially solved by using

the program generator for a majority of input programs and CULPRIT for most output programs. The contractor is provided a detailed design and development guide (Integrated Database Development and Design Guide, Version 2) which helps keep things on track.

As Figure 14 illustrates, the level of database development is growing. This factor must itself be considered when embarking on the database train. It WILL gather speed. As users become accustomed to improved and more sophisticated services which database can provide, demands will increase. It is the rare in-house staff which can cope with that increase and keep everything else going too.

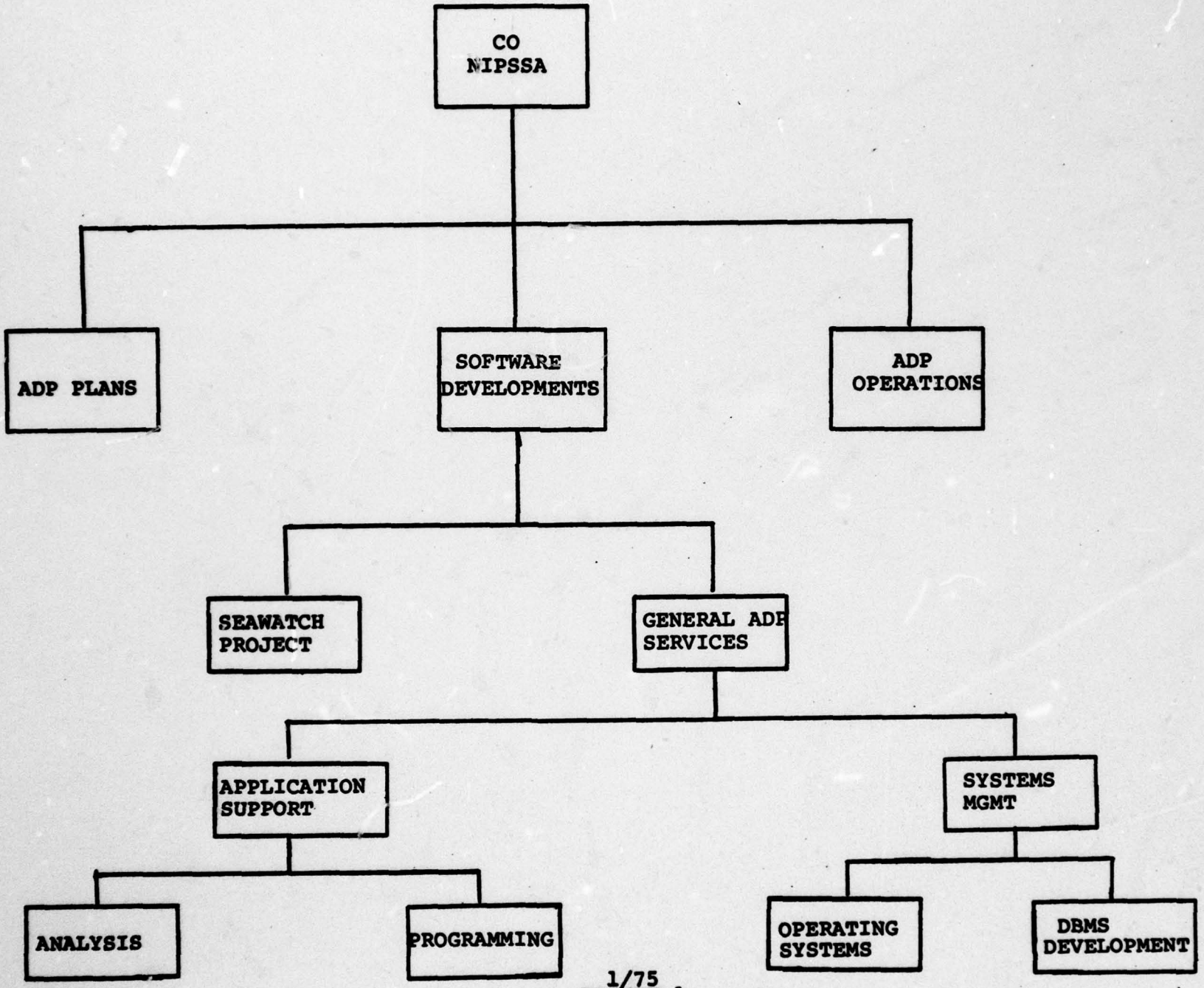
Last, but not least, is the hardware impact of database. Figure 15 illustrates only the main memory requirement trend associated with DBMS (and teleprocessing). The impact upon CPU utilization, disk storage requirements, and physical facilities can be expected to follow similar trends.

Of particular interest in this chart is the major REDUCTION in main memory requirements for applications software. This can be attributed to the modular input processing approach and the use of CULPRIT for output support. Modular software frequently runs with less efficiency than its tailored counterparts, but the savings elsewhere more than offset that disadvantage.

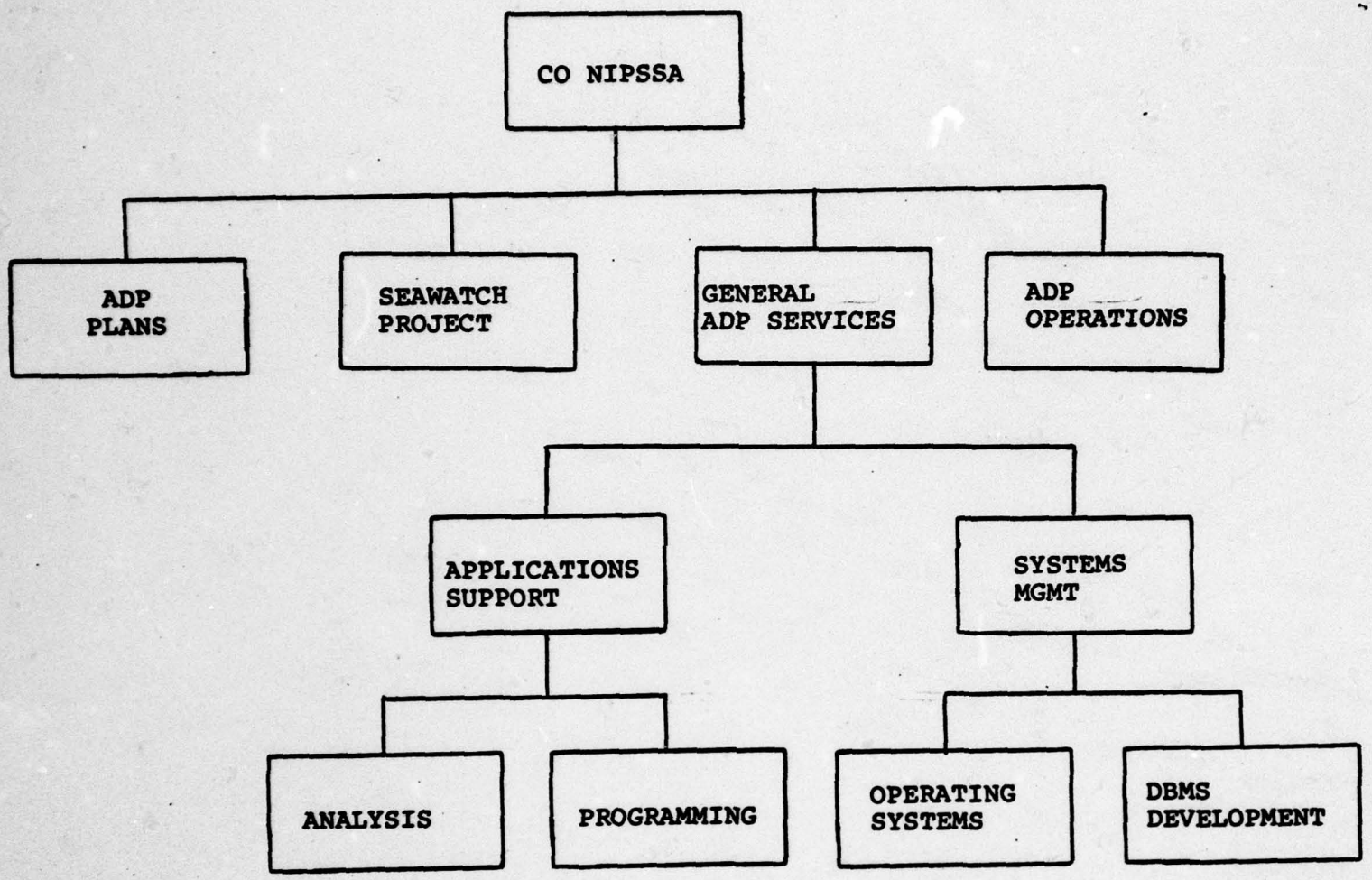
IN SUMMARY

Implementation of DBMS WILL have significant impacts on the management of any ADP organization which uses this functional tool effectively. Changes MUST be made in management philosophy, organizational structure, development techniques, and user interface techniques.

The ADP manager who successfully climbs aboard the database train, having matched his operation to the speed and direction of the train's travel, can be expected to have a productive ride to his destination. On the other hand, implementing a DBMS without such changes is an invitation to be run down by the train.

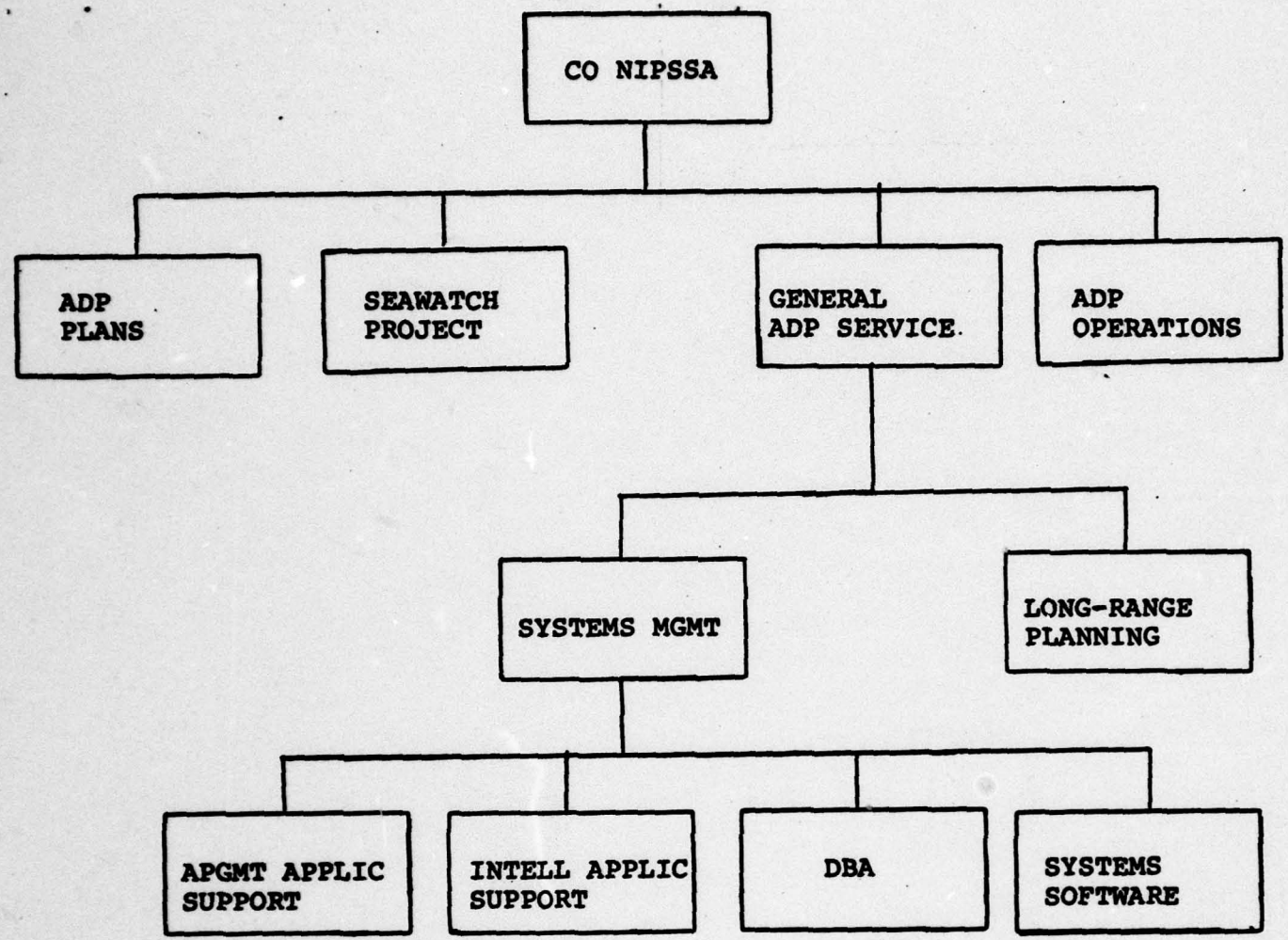

CO
NIPSSA
ADP PLANS
SOFTWARE
DEVELOPMENTS
ADP
OPERATIONS
SEAWATCH
PROJECT
GENERAL ADP
SERVICES
APPLICATION
SUPPORT
SYSTEMS
MGMT
ANALYSIS
PROGRAMMING
OPERATING
SYSTEMS
DBMS
DEVELOPMENT

1/75
FIGURE 1


CO NIPSSA
ADP PLANS
SEAWATCH PROJECT
GENERAL ADP SERVICES
ADP OPERATIONS
APPLICATIONS SUPPORT
SYSTEMS MGMT
ANALYSIS
PROGRAMMING
OPERATING SYSTEMS
DBMS DEVELOPMENT

6/75

FIGURE 2


CO NIPSSA
ADP PLANS
SEAWATCH PROJECT
GENERAL ADP SERVICE
ADP OPERATIONS
SYSTEMS MGMT
LONG-RANGE PLANNING
APGMT APPLIC SUPPORT
INTELL APPLIC SUPPORT
DBA
SYSTEMS SOFTWARE

2/76
FIGURE 3

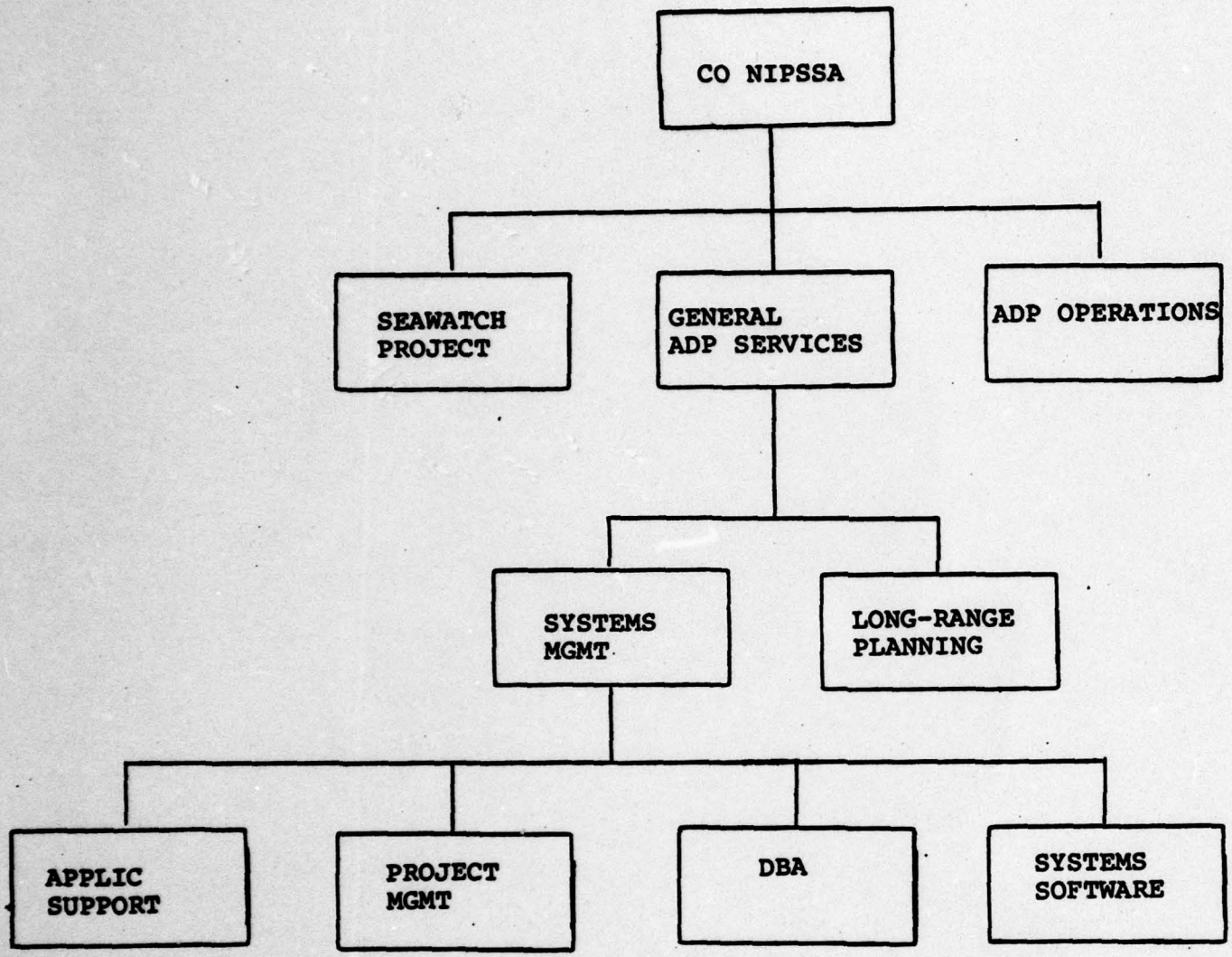

CO NIPSSA
SEAWATCH PROJECT
GENERAL ADP SERVICES
ADP OPERATIONS
SYSTEMS MGMT
LONG-RANGE PLANNING
APPLIC SUPPORT
PROJECT MGMT
DBA
SYSTEMS SOFTWARE

FIGURE 4
9/78

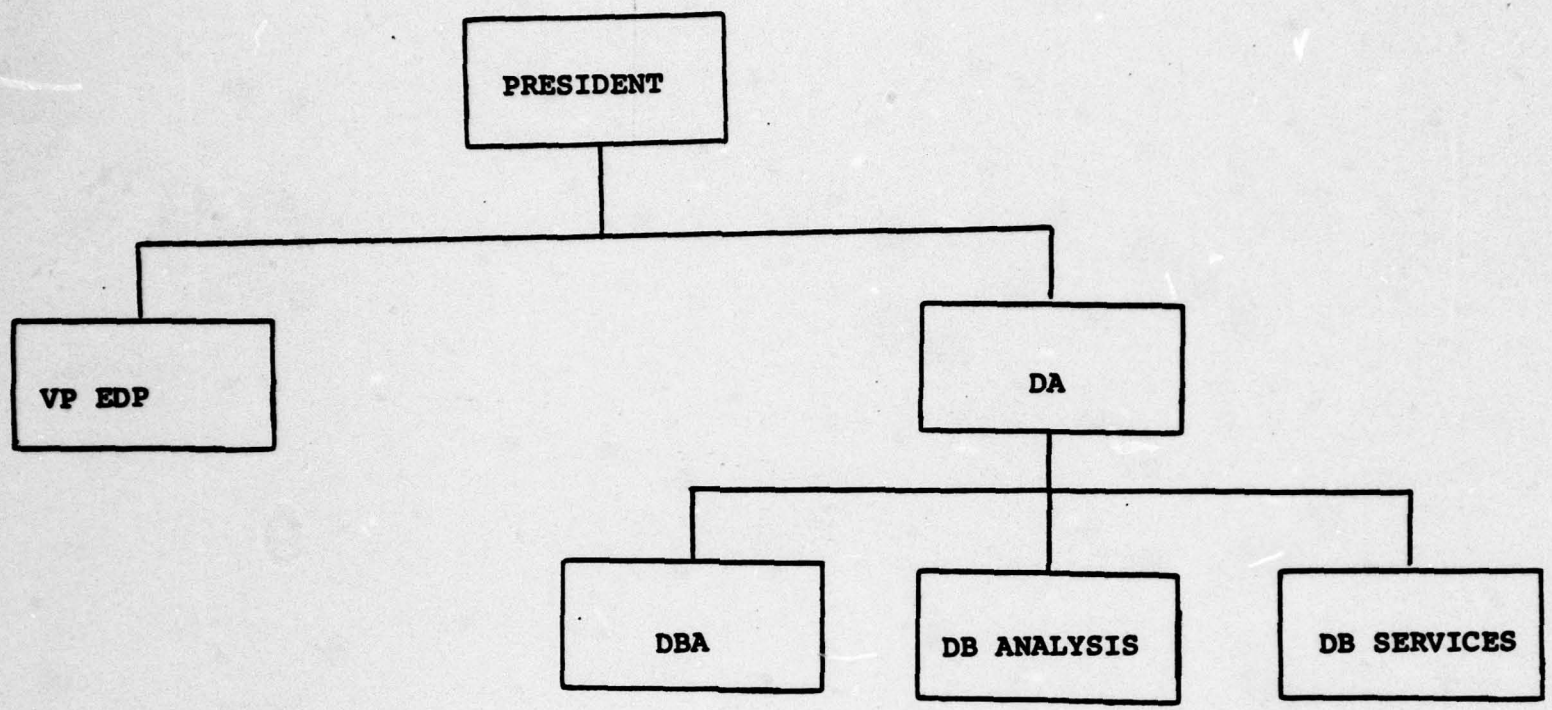
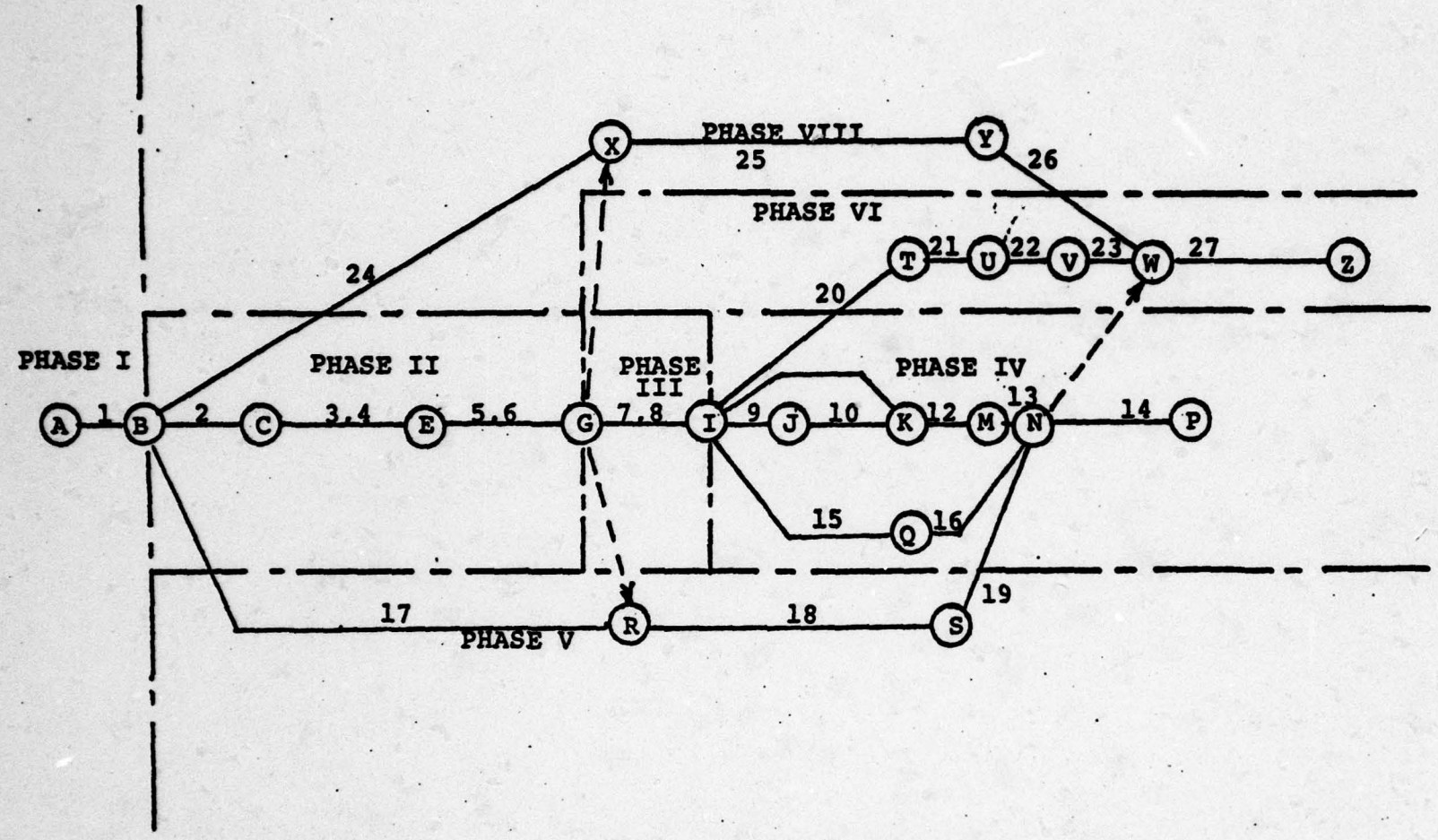
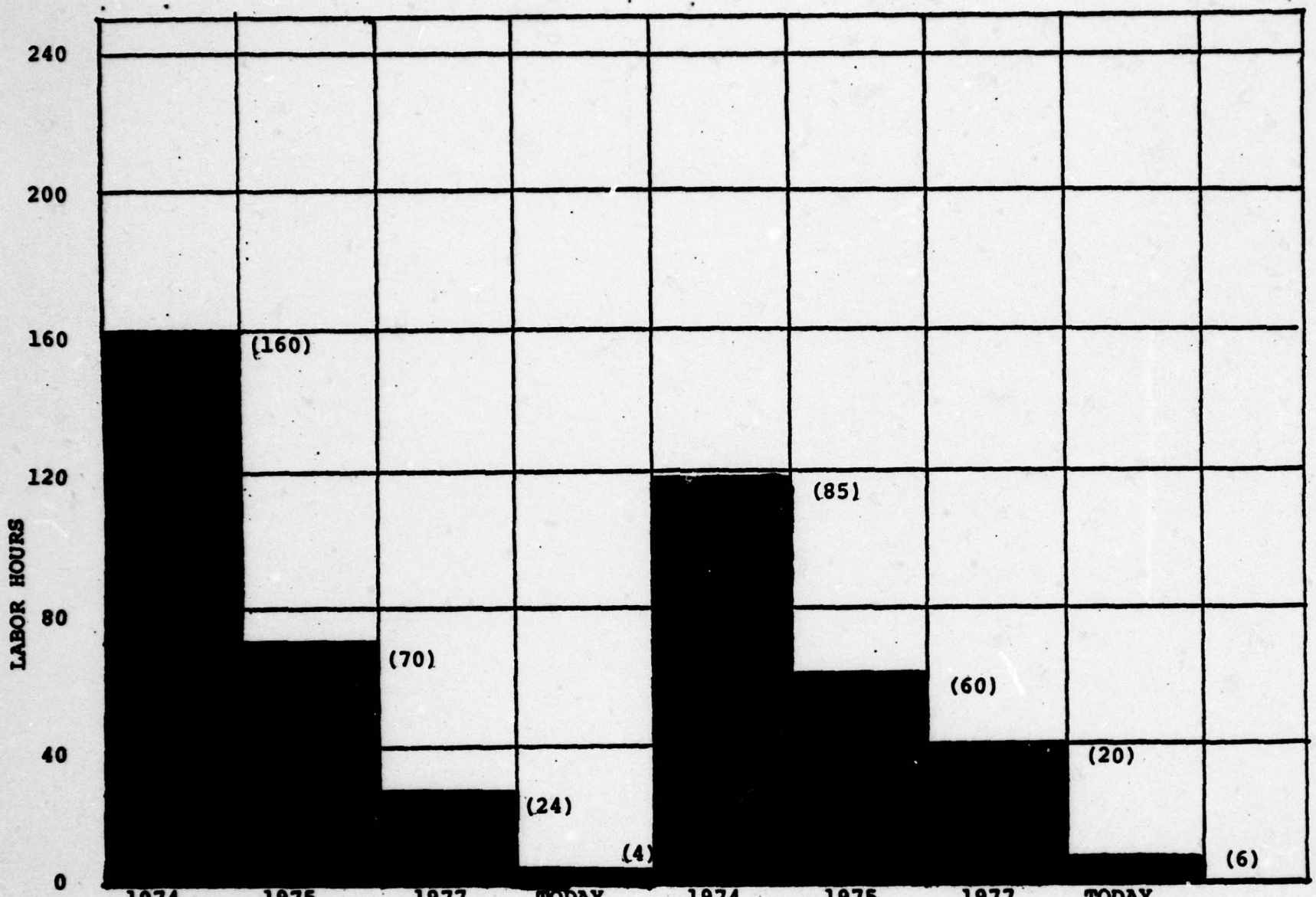


FIGURE 5
The "ideal" Data Administration
Organization Structure



DATABASE DEVELOPMENT TASK FLOW

FIGURE 6

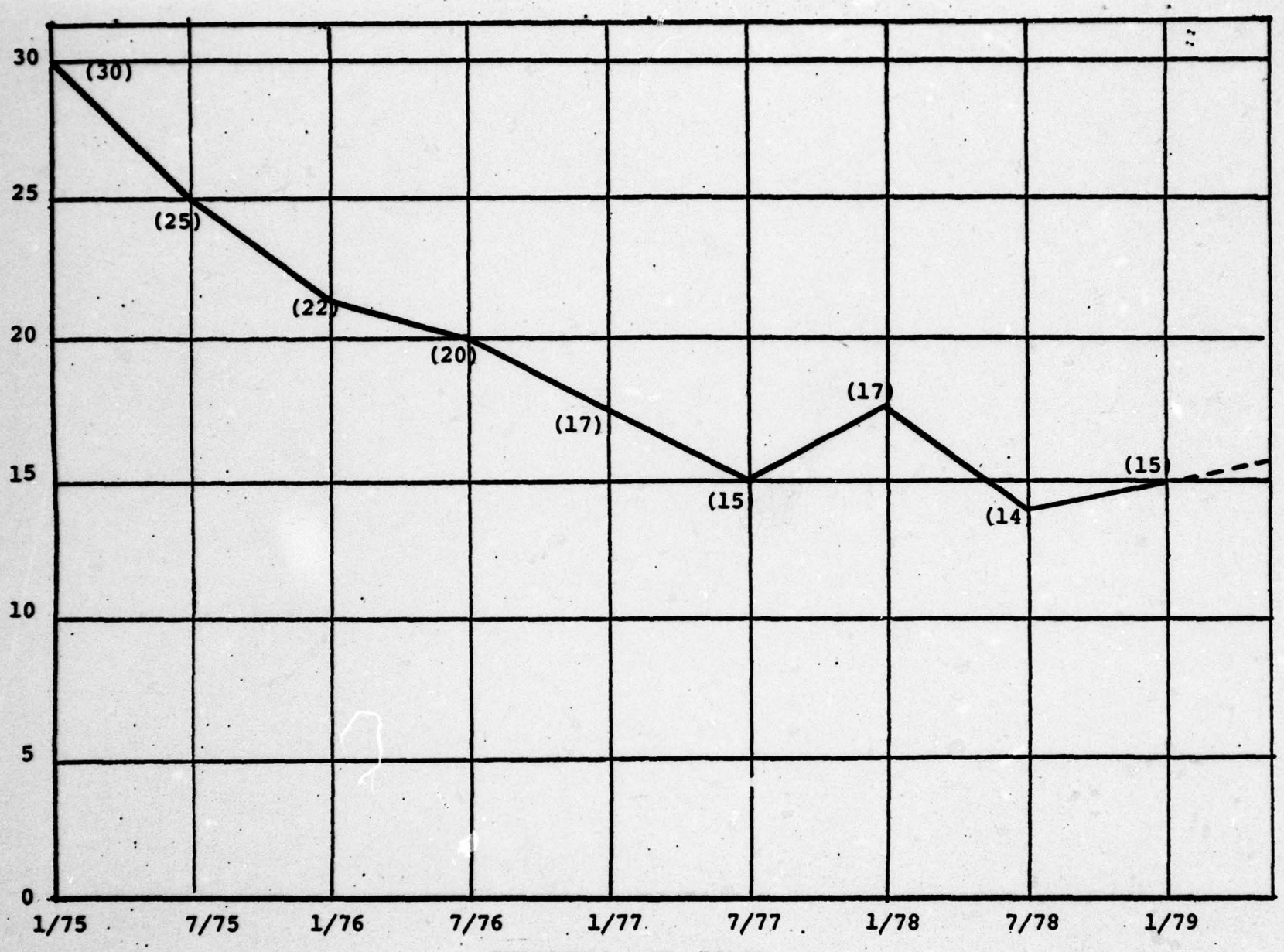


INPUT PROCESSING PROGRAM DEVELOPMENT

FIGURE 7

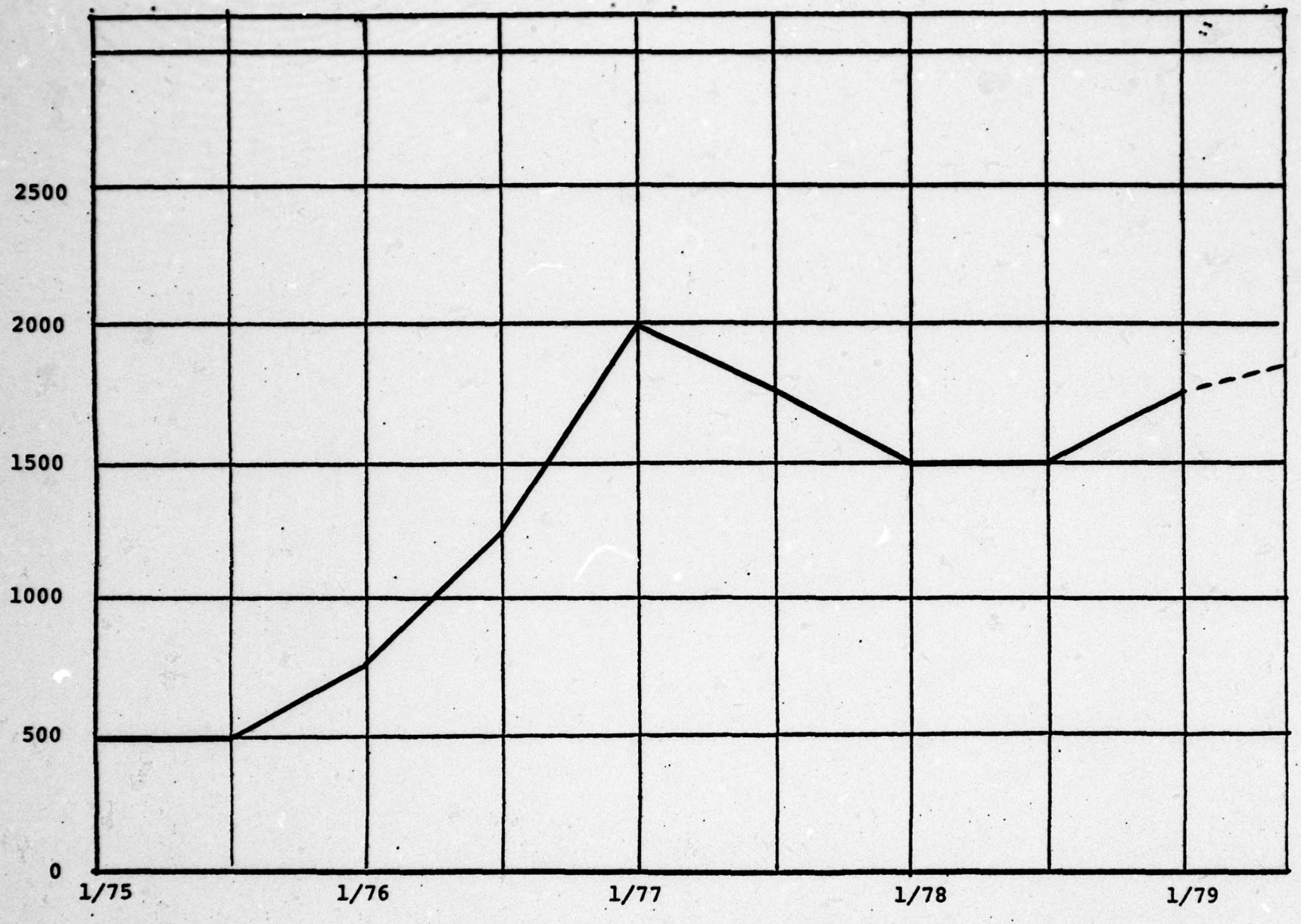
OUTPUT REPORT DEVELOPMENT

FIGURE 8



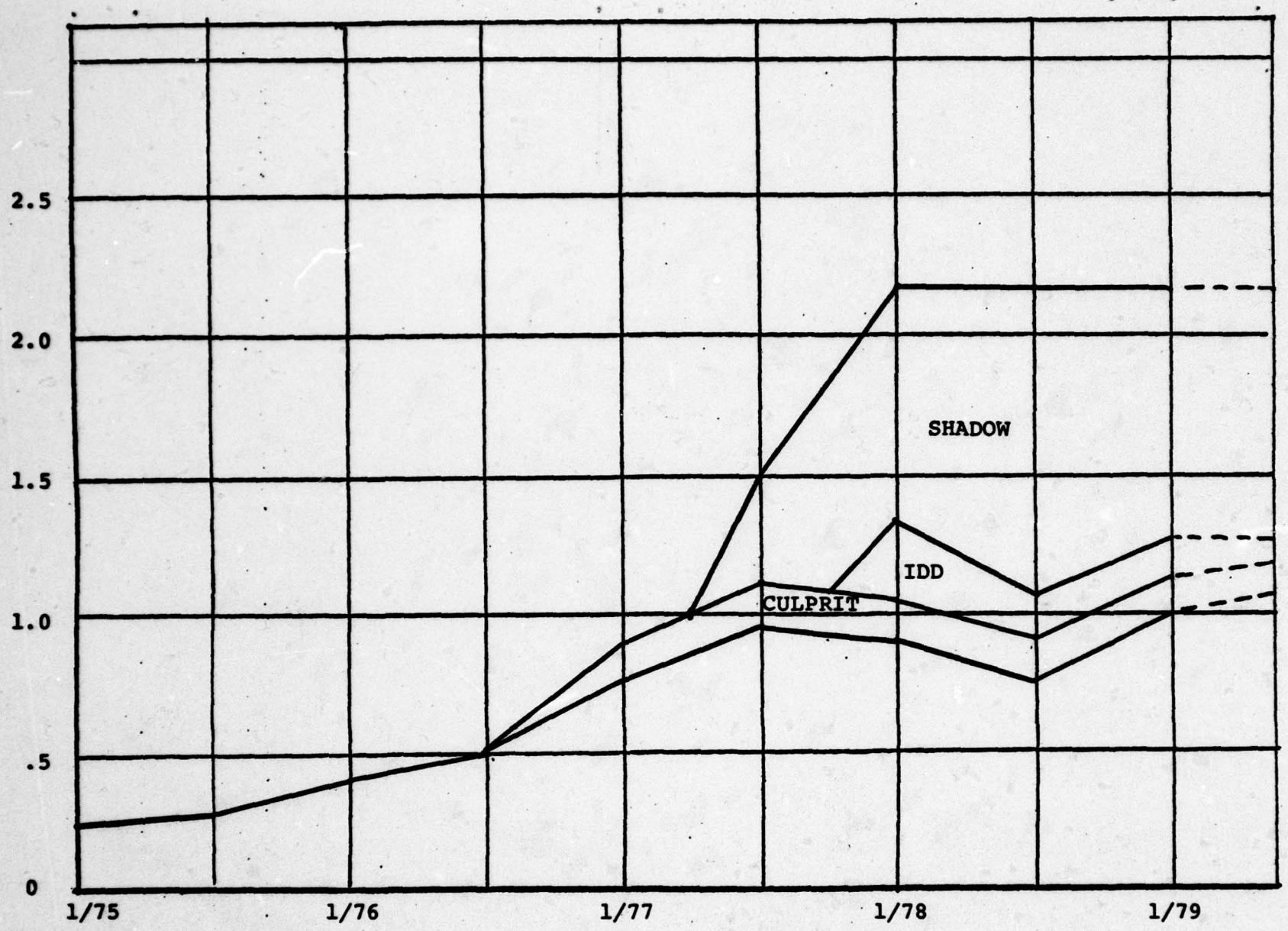
PERSONNEL ON BOARD
FIGURE 9

AVERAGE EXPERIENCE
FIGURE 10



TRAINING COST PER PERSON (LESS SALARY)

FIGURE 11



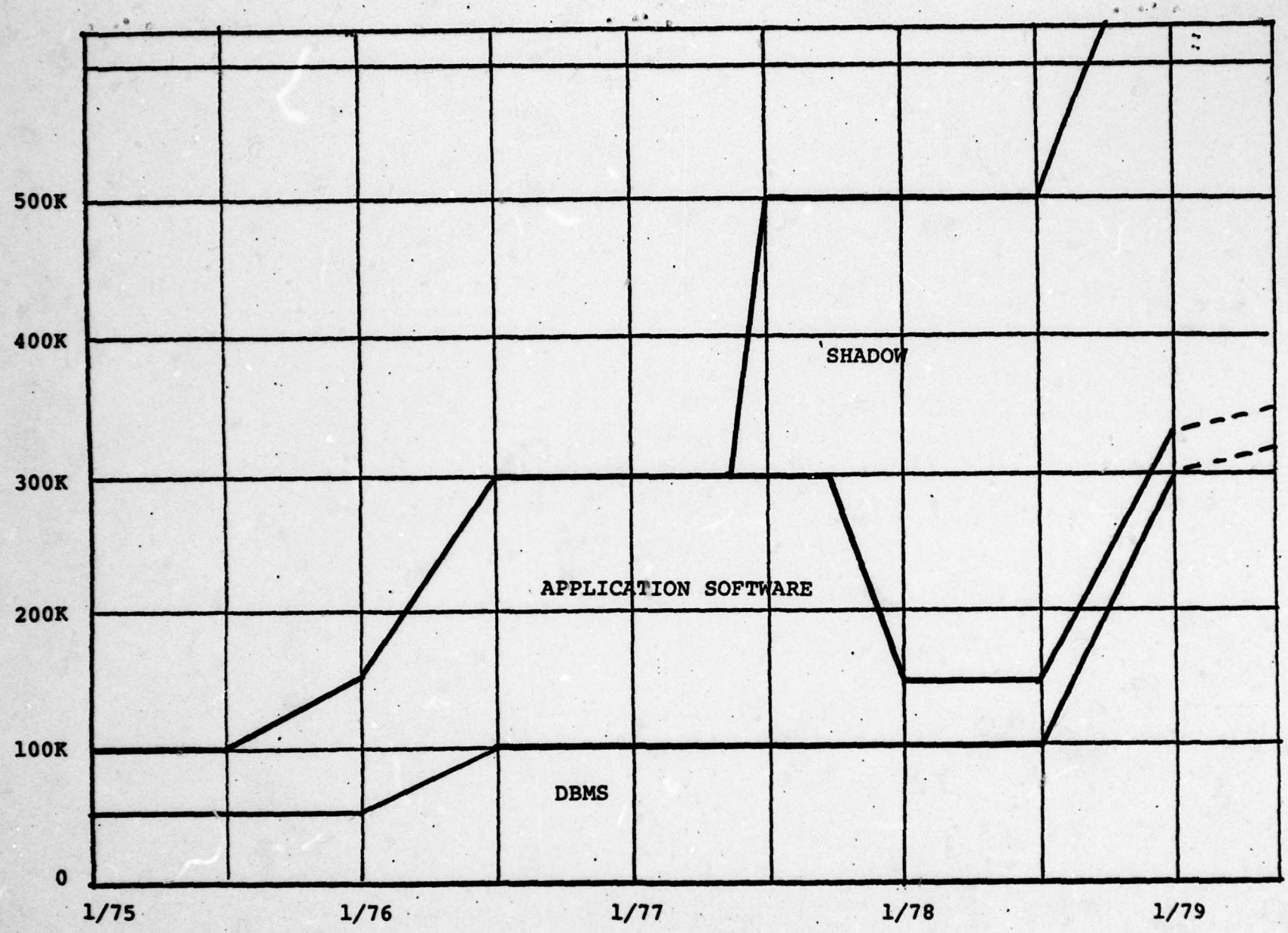
LEVEL OF SYSTEM SOFTWARE SUPPORT TO DATABASE

FIGURE 12

IN-HOUSE PERSONNEL RESOURCES DEVOTED TO DATABASE DEVELOPMENT
FIGURE 13

CONTRACT RESOURCES DEVOTED TO DATABASE SOFTWARE DEVELOPMENT

FIGURE 14



MAIN MEMORY REQUIREMENTS TO SUPPORT DBMS/TP

FIGURE 15