ACTUAL CONVERSION EXPERIENCES

The members of this panel reviewed their experiences under various conversion scenarios to identify the critical factors which led to success or led to delays and failure. The panel felt that the resultant list would benefit managers facing data base conversion projects. While it would have been noteworthy if some set of tools, methodologies and so on had emerged from the experience of the group which would make any future conversions easy and riskless, the panel found none. In the panel's opinion, a successful conversion takes tedious planning and careful execution, and in the current state of practice, no known panacea exists.

This panel will not try to tell you how to justify data base conversion, but will give its best advice on what to consider when approaching the conversion task. The panel began its considerations with the assumption that the justification had been previously determined. The panel agreed that its experience proved conversion justified. In fact, for some panel members conversion was unavoidable. Others felt they had demonstrated the value of conversion to DBMS, but with some reservations.

The panel consisted of practitioners, people who made or followed decisions to use DBMS techniques in actual situations and who, in most cases, had to suffer the barrage of consequences. The group reviewed only recent experiences, although several participants have been using the concepts and techniques since the early '60s.

The panel chairman was James H. Burrows, director of the Institute for Computer Sciences and Technology, National Bureau of Standards. Other participants were Edward Arvel, Data Sciences Group; Lt. Michael Carter, U.S. Air Force Data Systems Design Center; Joseph Collica, National Bureau of Standards; Elizabeth Courte, Bell Laboratories; Ahron Davidi, Blue Cross of Massachusetts; Ruth F. Dyke, U.S. Civil Service Commission; Halaine Maccabee, Northeastern Illinois Planning Commission; Steven Merritt, General Accounting Office, and Alfred Sorkowitz, U.S. Department of Housing & Urban Development.

The experience of the panelists consisted of two conversions from manual systems to DBMS, six from file to DBMS with one hardware change and two from DBMS No. 1 to DBMS No. 2, one also with a hardware change. Three vendors were involved: IBM, Univac and Honeywell. The DBMS packages consisted of IMS, TOTAL, DMS 1000, S2K and IDS I & II.

During the course of its discussion, it became clear that the panel and the people with whom the panel dealt saw DBMS technology, its promises and its threats differently.

One of the simplest views sees a batch system with an online access method to allow queries and, possibly, online data entry, such as a permanent operation using a leased national time-sharing network for access during the day with overnight update on owned facilities. One might use this operational mode as an educational step in the process of selling and orienting the user and technical personnel to the power and first-level details of DBMS, or as an intermediate step in moving from the DBMS to another.

Another view saw the DBMS as a "super access" method, promoting fast response and reducing the manpower costs of responding to new requirements.

A third view saw data base systems as the best way to permit ad hoc online access.

A fourth saw them providing multifile processing with minimal expenditure of technical talent.

However, all of the above were considered subsidiary to a view that DBMS controls the growth of information in support of the corporate business activities with less pain and suffering for all participants.

Many also see DBMS as a new sequence of buzz words and acronyms: schema, root segment, subschema, DDL, DML; a new mystique; a new barrier between the DP community and the real world. As an industry, we may yet learn that grist for the Ph.D. mill and true payoff to the business world are not necessarily the same. But, the boss knows it and suspects these new "academically sponsored" techniques. Who else invents language sounding like that above?

And as usual, the new technology threatens the current technologists.

Many of those selling DBMS emphasize the "total" commitment to the integration possible using a DBMS. This convinces management that DBMS, more than a new risk, means "total" risk. Who would make a decision so to "expose" the company? Many managers hear about the attributes of DBMS that make it seem a new technology that looks good for everything. But managers have seen other new technologies with similar histories. They share with DBMS the following traits:

- More variants seem to appear daily.
- No one seems to be in charge of standards, or even have a clue on how to start.
- What is available on the market is of uneven quality; ease of use, reliability, power and vendor support seem to be random variables.
- No accepted guidelines on "how to use" and "what to avoid" in the new technology seem to exist. This disconcerts managers.
- The technology conflicts with other forces. In DBMS the basic principle is that the corporation should consider its data as a central resource, not a private resource of each local subunit. At the same time, it is clear to the large national and international corporations that local authority is the only way to inspire local responsibility.
- Confusion exists over the fundamental purpose of the new technology. It is not clear whether the advocates of DBMS are technologists trying their best to meet requested/perceived needs or management specialists who see an opportunity to enforce central control. In any case, DBMS technology claims advantages for the user whether the corporation pursues centralized, decentralized or distributed responsibility, authority or data processing.

The panel found management's attitude about data processing shifting to concentrate on the data aspects. Data is not a private resource for local exploitation. Therefore, management must commit both to plan for and control the use of the data.

The first commitment, to plan for the use of data across the corporate body, is clearly a new job that is beyond the purview of any single department and also beyond the authority usually vested in the data processing department. Anyone trying to do central planning faces a struggle because each private domain will resist change.

Not only a new agent, but also a new function is needed for central planning: data base administration. Performed at two levels, this function may not be under a common manager. The first level is development/design-oriented. This group designs the data base, makes available the appropriate tools and utilities, analyses the usage of the data base(s) and restructures the data base. This group also establishes the procedures for the second level, the operation-oriented data base administrators.

These operation-oriented data base administrators must deal with the physical aspects of the data base. Such old concepts as allocation of storage, protection, recovery, dumps, verification and so on are performed in quite different ways under a DBMS.

Typical figures for the size of the DBA group may fall between four and 30. Only in very small installations is it a one-person job.

The panel unanimously warned against an overambitious project size for the initial data base application. This usually results in errors in cost and time estimates. The anxious user withdraws to a "wait-and-see" mode and announces changes in needs as the project progresses. The too-large project will probably be unable to cope with such changes, even if the original goals are successfully delivered. All of this leads to alienation of the users and managers and a distinct hesitation to continue with the new application technology.

The panel noted the availability of packages at many levels. However, knowledge of how best to choose and use a data base system is in short supply in any organization about to enter the data base world. Help from both consultants and suppliers is available and should be used.

The deliberations on planning identified a dichotomy between private industry and the federal government in both the processes controlling and the results accomplished.

Industrial/commercial practice, when making decisions on hardware and support software, requires a study of the choices, followed by negotiations with acceptable suppliers. The suppliers are then asked for their help in determining how to use their system. Users may insist that the DBMS and other packages come from an independent supplier of software.

This practice allows for a cooperatively developed proposal to be put before management, one in which each party to the proposal knows its role. More than one firm may be requested to make such a proposal. Outside consultants who are familiar with the various aspects of such a proposal may be used "to keep everybody honest." Such an evaluation may take nine months to a year, but significant preliminary design on how to use the to-be-supplied hardware/software system can be done.

In the federal government practice, whose major procurement dictum is maintenance of competition, the selection of hardware is predicated on a functional specification. For systems to be implemented in-house implies a hardware

functional specification to permit various subsystems of the hardware to come from different vendors. Specific details of support software cannot be specified if such specification gives special advantages to only one vendor or eliminates too many competitors.

At best, this leads to a watered-down software specification that may not be strong enough to support the intent of the internal federal government user. At worst, this leads to such a strong version of the specification that no one can supply an off-the-shelf software system. Testing such a system will take place long after selection. Systems lacking extensive field use are notorious for being unstable and poorly matched to the job at hand. Examples of such systems are the WWMCCS and the ill-fated ALS. It also means that nonvendor-developed support software is seldom bid by the equipment manufacturer.

This system of procurement produces a one- or two-year procurement process which requires a benchmark testing period oriented to some standard language. Such benchmarks cannot take special advantage of a vendor-specific operating system or DBMS.

After the hardware is selected, the internal software group must do a preliminary detailed design, using the specific hardware/software selected to determine whether the original requirements can be met. At best, this leads to an additional one- to two-year delay over private industry practice. At worst, it leads to a squabbling, contentious confrontation between two groups, with even a possibility of lawsuits, threats and ultimate abandonment of the original objectives.

Buying high technology from an adversary point of view rather than a mutually cooperating point of view leads inevitably to extensive delays and increased costs—if not to outright failure.

Note also that during the lag in the federal government procurement cycle, several things occur. The problem changes in scope. As noted in the keynote address, industrial growth in capacity required each year (stated not in dollars but in computation) grows 10 to 20 percent per year. The user gets discouraged and cancels, or worse, is given two more years to extract new promises from the internal DP manager. In addition, while waiting for hardware/software selection the internal DP team must bide its time, gamble on the winner and proceed concurrent with procurement, disband or assist the user in making larger plans. The first is wasteful, the second is risky, the third introduces delay and confusion after selection and the fourth leads to extensive over-commitment of both the to-be-acquired hardware and the internal manpower.

Thus, the panel found significant differences between private industry and federal practices. While the differences may have much larger implications, they also increase hardware costs for the federal government when buying a DBMS.

A way to avoid some of the difficulties described is for the federal agency to create a truly functional specification in the user's terms and to ask for a competitive turnkey total system. This will eliminate third-party sources, since the total system is to be procured and supplied by a single source. However, upgrading the system to handle additional growth requirements would require, at best, an interim upgrade and, at worst, a totally new procurement process. This would be anathema to a concern with a profit motive. Simply stated, cost-conscious organizations would not consider the federal government's methods to be good business practice.

When installing a DBMS for the first time, management, user and service supplier must get their feet wet in a quick and successful project. This argues for phasing implementations into small packages deliverable in, at the most, four to six months. This allows each group to develop confidence, give feedback and adjust its plans and expectations to match better the tasks at hand and to feel, in general, comfortable with the changes.

Installing the first DBMS will result in changes to lines of authority and responsibility, some real and some apparent. Do not let the new data base administrators be stymied by the old warlords. Fiefdoms exist and must be continued, but the data base administrator is responsible to the organization as a whole and needs the necessary authority and management support. Of course, this new operating mode will require new responsibilities and corporate approval before old, negotiated responsibilities can be discarded.

One way to get assistance in promoting acceptance of the data base administrator is to keep the communication lines open. This enables all to observe the changing environment, to get frequent statements of management support for the data base administrator and to reaffirm the resolve to continue and complete the desired conversion.

It may come as a surprise to old hands in the business that conversion to data base environment from either a current file system or a manual system will each require a fault-tolerant or "forgiving" mode for data base generation and update. Even from automated files, there may well be illegal values, voids and so on in the file. This could cause a significant number of errors to be found by the edits when building the new data base, especially if internal DBMS pointers are value-dependent.

Your users will need help. Do not "over-automate" on the initial change from manual files. Use the current data in its current forms to meet today's needs. Get it into the data base and clean it up as the data is needed for new applications. Cleanup can be a significant and unplanned-for activity. Accept the dirty data and give some service of value. Do not take the position that it's their fault that the data base cannot be generated. Help them.

For those converting already-automated systems, keep a

parallel backup for each full-system conversion until the new system is solid. This may be three to six months. Try scanning a file for range of values on an item; illegal or nonsense values are frequently present.

The panel regretted the lack of common terminology in the data base management arena; indeed, it appears that every new development of any size brings with it an opportunity to create a plethora of new names and verbs to distinguish minor variations. The underlying fundamental concepts should be given standard terminology and variants should be clearly explained and justified.

Conversion tasks would become easier if each data management system had the same functions available, or possibly a basic subset of some superset. Common functions with common definitions would create the same result for the user, even though the implementations varied. This may be both unachievable and undesirable, given the several differing fundamental file forms. However, it may be possible within classes of data base systems.

A third useful area for standardization is a micro-language (atomic verbs) in which the functions (commands) of a data management system can be described so the detailed specific actions of each command are obvious. This would make definitions more clear and precise. It would facilitate comparing DBMS and it would facilitate reimplementation of a DBMS or emulating one DBMS with another, an action which might be needed to facilitate changing hardware and DBMS.

The panel saw the need for tools to assist conversion from one DBMS to another. Unfortunately, they might have to be dependent on a particular situation, but converting one DBMS file to another should be possible without going back to transaction mode or card-image mode, the most prevalent way of generating a data base.

Another technology development needed is the creation of a model-independent data description that could be automatically mapped into any data description language.

The rapidly changing field of data automation insures that opportunities will always be available for transition of application software. This transition may involve evolving from a non-DBMS environment to an exclusively DBMS arena, changing from one hardware environment to another through a common DBMS, changing from one DBMS to another on either existing or new hardware or entering the distributed processing area.

Perhaps the most traumatic of conversions involves the initial move of an organization into the DBMS world. For the traditional application programmer, the DBMS looms as not only an unknown world, but also as a threat to job security. The first task then is to provide the threatened programmers with the proper training, not only on the specific DBMS to be used, but also on the advantages and disadvantages of DBMS as a whole. Of great utility in the transfer of people is on-site, vendor-provided training.

Groundwork for the move, however, should initially be laid through commercially available courses dealing with data base systems independent of a vendor implementation. Initial training should begin either before or during the DBMS procurement cycle so present personnel will see the DBMS for its utility and not as a threat. Furthermore, thorough conceptual training will insure greater utility from the DBMS vendor's on-site training. Care must be taken to insure that the vendor's course includes a practical exercise to demonstrate the advantages of DBMS, specifically in the areas of interfaces, DDL and DML.

The wise planner will quickly recognize that the most important resource for training people is time. If possible, time should be set aside following vendor training for application programmers to experiment with the procured system before requiring production programming. This exercise may involve design and programming for data conversion needs.

The transfer of data from independent files to DBMS control can require extensive resources. The process is usually a download by stand-alone application programs, then upload through DBMS-dependent application programs. Although the concept sounds easy, the volume of data as well as variety of storage formats, devices and locations can, and do, drive the cost of transition to very high levels. The solution to this problem lies mostly in the continued efforts of managers to keep current their documentation on files and their uses. This documentation may take the form of written narratives and file format charts, but perhaps the most effective tool is the data element dictionary/directory. This automated system can provide managers with current information as to the type, size, location and usage of data. Although used most extensively in coordination with DBMS, the dictionary/directory can insure that managers and those responsible for the inclusion of data in a data base are provided a complete view of the agency data as a whole, as well as usage of that data by application programs.

In addition to these tools, research by Jim Fry of the University of Michigan and Arie Shoshani of SDC [see the Conversion Technology panel report] in data translation and conversion promise future tools for describing source data files and providing translation to some target format. Emphasis should also be placed on determining a common interchange form for the conversion of data between hardware types.

Moving from a non-DBMS to a DBMS environment involves, in most cases, a matching of like functions in many independent application programs to shared functions in the DBMS. Consequently, this movement involves removing code from the application programs and replacing it

with code to interface with the DBMS. The bulk of application-dependent code should remain the same. Capabilities such as backup/recovery and physical data structuring will be moved as a whole to the DBMS. In the DBMS environment, no capabilities are lost to the application programs—only gained.

No tools are currently available to aid in this transfer of capabilities, but all DBMS share a common repertoire of functions to include facilities for data creation, update and deletion. These capabilities, however, vary depending upon the underlying conceptual model of the DBMS. Standardization of DBMS capabilities and syntax could allow for a more complete predesign effort through knowledge of system features.

As mentioned earlier, conversion forces management to take a much tighter control by establishment of a centralized function known as data base administration. Within that function, procedures are further established to deal with data integrity, accessibility, confidentiality and miscellaneous control. The primary tool of the administrator is clout or having a position in the organization with the authority and responsibility to insure compliance with established procedures. The dictionary/directory is also used by the administrator to aid in the design, implementation and maintenance of the data base.

Perhaps the least explored area of conversion and need tools involves the migration of applications from one DBMS to another. This conversion process is second in impact on users only to initial entry into a DBMS environment. The training of personnel on the new DBMS is simplified somewhat by a previous exposure to the DBMS way of doing things. The application programmer, however, will be required to view the data in a new manner, especially if the move is from one conceptual model to another, such as from a network to a hierarchically-oriented DBMS or vice-versa.

Training will be the primary tool in conversion, including both vendor and follow-on in-house training sessions and exercises. For conversion between certain DBMS (CODASYL-like systems, for example), automated tools may be used for direct-syntax transformation in both DDL and application programs. Conversion, however, between systems which do not share a similar syntax or mode of interface, such as TOTAL to IMS or IMS to DMS-1100, will require extensive training to learn the new data manipulation language (DML), DDL and interfaces.

In migrating data to the new environment, no new problems will be encountered which have not previously been discussed. The download programs should, however, already be developed for such purposes as data base archiving (dumping).

The transfer of capabilities provides the greatest impact on the conversion between dissimilar DBMS. DDL presents the first difficulty in conversion. To implement the new data base, some description of the new data base structure must be developed in the target DDL. Automated translation tools can be developed if the source DDL is extensive. Manual translations, however, seem more appropriate due to various changes in the way in which DDL may be interpreted; DATA SET in DMS-II, for example, is not the same as DATA SET in IMS.

A DML, as mentioned earlier, offers the most advantages to automated translation when the interface syntax is fixed (CODASYL's DML). When the interface is purely through dynamically built character strings (IMS, TOTAL), automated conversion can only replace CALL statements of one format to CALL statements of another. The DML functions, although common (GET, PUT, CREATE, DELETE, UPDATE), also vary in interpretation. An example: FIND in a CODASYL-like DBMS only locates a record as stated in the record selection expression and a subsequent GET must be issued to load the data into the issuing program's work area. A GU (get unique) in IMS not only accesses a record but also requires the segment selection argument (SSA) to find the appropriate record.

Therefore, the capability of a single DML command may not be replaceable by another single command of the new DBMS. Conversely, several commands which span both DDL and DML (record selection expression/set selection) in a CODASYL-like DBMS may be replaceable by one command (GU W/SSA). Other commands of the source DML may not be duplicated in the DML of the target DBMS. Apart from the DDL and DML capabilities, more basic differences may exist.

Shared functions which DBMS supply are often quite different. Where one DBMS may supply item-level locks, another may make only record or data base locks available. Access controls, recovery and auditing are just a few of the support capabilities which vary between DBMS, even those which implement the CODASYL specifications. The change in capabilities, therefore, is indeed the most evident conversion problem in migrating applications between dissimilar DBMS.

The transfer of procedures will usually be transparent to the application programmer, but will impact heavily upon the DBA. Because of the differences in capabilities and architecture of DBMS, the DBA will be forced to adapt fastest to the changing environment. The DBA must immediately recognize differences in the operation of the new DBMS and take steps to avoid turmoil. Procedures must be changed as little as possible in external appearance. The application programmer must interface with the DBA in the same manner as before. The DBA, however, must determine and apply new procedures to insure maximum system efficiency based on the information passed across this interface. The DBA's primary tool in this environment will then be training and any available consulting services.

The most frequent motivation for conversion is caused by an impending upgrade in hardware from one manufacturer's equipment to another or between nonsoftware-compatible lines of the same vendor. Within this environment, change can extend from simply migrating applications from a DBMS on the current machine to the same DBMS on the target hardware (TOTAL on IBM to TOTAL on CDC), through changing DBMS between machines (IMS on IBM to DMS on UNIVAC 1100) and, finally, to initial implementation of a DBMS environment on the new computer. Two of these situations (DBMS 1 to DBMS 2, non-DBMS to DBMS) have been previously discussed. The additional impact of changing hardware, however, adds a new dimension to the training problem. Migration of data also is a bit more difficult in that some form must be established in which to download data for subsequent upload on the target machine. Transfer of both capabilities and procedures apparently are not impacted or complicated by this transition.

The additional transition of DBMS on the current hardware to the same DBMS on the target hardware begins to show the utility of standardizing both interfaces (end-user facilities) and capabilities. Whether the DBMS in question is CODASYL-like or not, costs are significantly reduced when syntax and capabilities remain constant across hardware types. Examples of systems which provide these transition opportunities include TOTAL, SYSTEM 2000 (non-CODASYL) and IDMS (CODASYL-like). Applications which utilize these systems can concentrate on more operating system-dependent transitions (JCL 1 to JCL 2) because of the realtively small changes required in DBMS-oriented syntax. Because of the vendor-enforced (or de facto) DBMS standardization across machines, automated tools can greatly aid transition and may be as simple as filtering source programs and recompilation.

With cheaper system communication costs and better cost/performance ratios of the minicomputer (some supporting virtual memories in the millions of characters and most, if not all, offering large-scale peripheral disk storage), the use of data base software can be relocated from centralized large-host environments to mini-based decentralized arrangements of data processing facilities. Conversion from the traditional centralized mode to distributed data base processing is fraught with both technical and management perils.

The mini-based DBMS generally does not offer the diversity of facilities available with data base systems centrally resident in larger hosts and, in fact, may constrain the design of the application systems planned for dispersal. Replicating the application software, data base structure (DDL) and data base software may require a severe reorientation of existing DP management philosophy. Management and control of multisite operations, DP standards, application development and data base administration will be made more difficult. Conversion of data previously centralized requires that some intermediate data form be created for downloading appropriate subsets of the old file(s) and subsequent uploading on the distributed hardware. Standardized interfaces (DDL, DML and end-user facilities) and common DBMS capabilities will reduce the management problems encountered during the design, development, conversion and operation of distributed data-base implementations (even if data base management software and minicomputer hardware differ from site to site).

The problems of conversion from a centralized mode of DBMS operation to distributed data base processing are somewhat eased by the presence of experienced and knowledgeable personnel at the beginning of the conversion, but are still of concern. Aside from the difficulties associated with the migration of existing application code (dependent upon the existing business functions to be dispersed, degree of transferability of the programming languages presently in use, limits on size of load modules and so on), further problems surface if dissimilar data base logical structures (network, hierarchical relational and so on) are encountered.

Conversion to a distributed data base structure that is similar to the existing centralized structure should be an easier task. DDL differences may exist, however, between the existing and target data base systems as well as the obvious job-control language differences. The target (distributed) data base software may not offer the facilities used (or planned for use) in the existing centralized applications, and support of user business processes may thus be reduced. The problems noted in the conversion of centralized nondata base applications to the decentralized data base mode remain unchanged for the migration from centralized to decentralized data base modes.

The panel concluded that the single most important guideline to offer a group about to embark on a system conversion was to use its best management techniques. It is a technically difficult job, like most large software system developments, and one must apply all well-known software development methods.

The discussion of the various conversion scenarios revealed the fact that several management practices had been utilized in all of the successful conversions which, though common across all data processing environments, were of particular relevance in the data base environment where problems and errors in establishing policy proliferate across all applications utilizing the DBMS.

These considerations were deemed important:

- Analyze all possible file organizations. Do not assume that the most efficient way is going to be a data base. Some applications can perform better using sequential files such as tapes or other access methods such as index

sequential.

- Analyze the final stages of the conversion. How will you convert to production? Will it be (or can it be) converted module by module, transaction by transaction or will it be necessary to "push the button" and convert to production all at once?
- Develop a representative model of the company's business for design purposes. This model should provide the basis for the design of the data base which will be in production.
- Determine the degree of security and integrity required for the operation of the data base. Who will have access to certain information? How will unauthorized use be prevented? How will you recover or reconstruct the data base in the event of a software or hardware failure?
- Determine the space requirement for the production data base. Determine if all the data on the file is necessary and is being used. Check for redundant data. (It is not always bad to have redundant data, especially if it will improve retrieval time.)
- Build a small version of the data base and test for deficiencies in the design or in the modules. The testing must be thorough and cover all facets of the company's business. User involvement in this stage should be heavy, but it should not dictate how the data base is to be designed or what access methods must be used.
- Determine whether a data base administrator is needed for your organization. The DBA will be responsible for the security and integrity of the data base (including recovery and backup) and act as an agent between the user group and the data processing group.
- Determine the type of support you will need from the vendor. Will classes be given on data base technology? Will the vendor be readily available when a problem arises? Contact various users in your area. Attend a local user group meeting and find out the types of problems and solutions that the group has come up with. Especially talk to users in the same type of business. Valuable information can be obtained and you will not reinvent the wheel.
- Determine how eager upper management is to undertake the task of converting. If it supports you, the task will be easier and more efficient.
- Look into the future. Determine whether the present design of the data base is the most efficient one in case new applications are introduced. Do not be afraid to redesign if it proves to be more efficient. Provide for a purge of unneeded data. Reorganize the data bases on a regular basis to reclaim unused space or to compact the data. One very large data base may not be feasible.
- Determine whether to use standard vendor-supplied software or to develop your own.
- Once a decision has been made to convert to a DBMS, check and recheck the contract to be signed. Have a contract attorney study the agreement.

In successful conversion efforts, special emphasis must be placed on establishing procedures and policies and insuring they are followed. Examples of these procedures and policies include internal standards, data base administration and explicit goals.

The probability of a conversion succeeding is directly proportional to the degree of preparation. Planning efforts involve:

- Establishing current baselines and identifying perceived inadequacies.
- Defining requirements or specifying what the objectives of the conversion are.
- Enumerating as many alternatives as practical and determining their appropriateness in the target environment.
- Establishing feasibility in terms of costs, people and time.
- Describing milestones and benefits.
- Receiving and documenting a management commitment to a specific alternative.

Planning efforts must begin immediately upon management commitment and include:

- Organizing for the project.
- Defining support software requirements.
- Evaluating available systems (packages).
- Describing selection criteria.
- Selecting and procuring software.
- Building a well-defined implementation plan including staffing, training, detailed design and implementation strategy.
- Seeking final plan coordination and approval.

The panel recommended several actions:

- Establish review points. When describing a plan, care must be taken to insure flexibility to support changing/overlooked requirements. Points must be scheduled for reviewing and updating as required. Decisions must also be made as to the appropriateness of proceeding with development.
- Involve end-users. The data processing department must be extremely careful not to ignore user input to the development process. No one knows the functional requirements which must be met better than the end-user.
- Keep the scope reasonable. The scope of the project must be dictated by practicality. Attempt only as much as is attainable within well-defined time frames and technology. Once the scope is defined, stay within its bounds.
- Do not stifle prototyping. Modeling is perhaps the best method available for trying out ideas. Encourage validation of concepts through prototyping.
- Shift responsibility to where the expertise exists. Always insure that responsibility for achieving goals is placed

where both the ability to understand and accomplish them exists.

- Phase the implementation. Within an overall context, insure that total system conversion is planned and executed in attainable increments.
- Be wary of entanglements. System conversion planners must always review commitments with the goal of avoiding crippling dependence upon manufacturers, proprietary packages and maintenance agreements. Recognize the cost/benefits of new technology. Quantify both the advantages and disadvantages of being the first to use new technology.
- Get adequate management commitment. Make sure management knows beforehand what resources and time are required, what has been available and what progress, if any, has been made. Do not let them walk away from it. They are key players.
- Select initially an important but tractable portion of the ultimate and have some results in four to six months. This forces an early and realistic review. One also hopes it demonstrates the new capabilities, emphasizes the ability of the implementors to handle the new technology and provides better insights into both the technical and operational (people-oriented) problems the organization will face.
- Introduce your technical staff to the new technology. Do it first when you are studying whether to go data base, but also immediately before and during the initial implementation. Sometimes six months to two years pass from initial study to start of implementation. Further, training for the initial decision process is usually "book learning" and quite stale by implementation time.
- Orient and educate your users. Next to management, this group is the key to success. It must feel comfortable with what you say. That means it should have a role in deciding what it gets and when. In addition, users must prepare themselves for the new modes of operation.
- Keep the user group on your management review team. It can keep you out of trouble. Users can change their priorities and needs to fit your capabilities to deliver. They can help you resist pressures for early and extra delivery since they, too, want and need a quality product. By keeping them in the loop, you risk less chance of surprising them and evoking resistance on delivery.
- Set up a central data base control group. Useful in any big system or series of related systems, it becomes essential in a data base-oriented organization. Such a group may have several levels with differing managers, but these groups must be coordinated. There is a policy level, an implementation level and an operational level. The implementation group does most of the administrative and design work involved in bringing into being and providing for the efficient structure of the data base(s). The operational level is responsible for the operational integrity of the data base and must take the actions for restart/recovery in addition to any periodic initialization/dumping or consolidation of the actual data base.
- Implement a data dictionary/directory. As a key item common to all systems, it is needed to make the data base a corporate rather than a private entity. Because the data is to be shared across groups which are not responsible for the data, it is essential that the meaning of each data item be documented. A data item, besides having a name, a form and values chosen from some well-documented set, has several meanings. One is the English description; another is an operational description of how the value is determined; another is how the data is commonly used and by whom. All of these must be known to avoid confusion and to avoid having data misused.
- Ask for and accept help from the vendor of the data base system. If you are also changing hardware at the same time, put pressure and some of the risk on that vendor. Good advice must be available during your learning period. Consultants are useful, not only initially, but also during design and implementation time.

The most significant finding: The ingredients for success are management commitment and discipline, skilled people, clear roles and lots of cooperative conversation between the parties involved. Not a new discovery, but still significant—and more essential than ever at this stage in DBMS evolution.

The three user experiences which follow illustrate the scenario of converting from a file system environment to a data base environment without a change in hardware resources.

Before conversion No. 1, studies were conducted to analyze the file environment. Problems identified by the functional user and data processing departments included prohibitive maintenance and enhancement costs, slow implementation of additional user requirements, heavy data and processing redundancies and lack of data integrity.

After it was decided to adopt a DBMS, a consulting firm prepared a conversion plan to execute tasks leading up to but not including the conversion of data or application systems. The consulting firm proposed the following procedures:

Step 1. Evaluate the Present Information System
- Information Flow Analysis
- Current DP Analysis
- Current Reports Analysis
- Status of Present Information System for Data Base
- Draft of Data Base Administration Responsibilities

Step 2. Define Data Base Requirements
- Service Analysis Development by Report

Service Analysis Development by Function
Data Dictionary Development

Step 3. Develop Initial Data Base Architecture
Distribute Data Dictionary Elements
Distribution Optimization
Architecture Description

Step 4. DBMS Package Evaluation and Recommendation
Architecture Mapping
Support Features
Secondary Features
Recommendation

Step 5. Application Redesign
Design and Program Documentation
Structured Programming Analysis
Implementation Controls

Step 6. Configuration Analysis and Evaluation
Current Load Analysis
Future Load Analysis
Simulation Modeling
Simulation Analysis

Step 7. Implementation and Conversion Plan
DBMS Installation Planning
Data Conversion Planning
Application Conversion Planning

Step 8. Personnel Development and Training
Assess Available Talent Levels
Develop Formal Training Plan

The consulting firm provided project direction and DBMS expertise. Company personnel assigned to the project received practical training while preparing for planned conversion of application software and data files. Some points should be noted about the tasks prepared for the data base system plan:

- The evaluation of the results from the first four tasks of Step 1 was used to support the data base decision.
- The data base administration function was established following a draft of responsibilities prepared in the fifth task of Step 1.
- The initial data base architecture described in the third task of Step 3 was package-independent.
- The DBMS package recommended in the fourth task of Step 4 was influenced by constraints imposed on the evaluation process. The influences included the need to interface with teleprocessing and the ability to install DBMS with an initial or pilot application system conversion in just four months.
- The application conversion plans in the third task of Step 7 identified functional redesign requirements resulting from the service analysis conducted in the second task of Step 2 to correct deficiencies in existing software and to benefit from the facilities offered by the DBMS package.

The recommended DBMS is installed and the conversion and implementation of the pilot application system and its data files to data base is completed. The conversion effort was successful but not without its problems. The functional user organizations feel the benefits of DBMS in the areas of performance, data integrity and responsiveness on the part of the data processing department to change requests. However, further implementations to DBMS are suspended due to a redefinition of priorities outside the data base system plan.

Several factors contributed to the success of the conversion. Management demonstrated commitment by the funds made available for consultant services, by the investment in planning and by the establishment of a DBA function. This commitment was instrumental in the responsiveness and cooperation that was sorely needed for data processing personnel and users alike when the data base project was undertaken.

The establishment of a DBA function in the data processing organization invariably creates a political struggle involving the DBA, the manager of systems and programming and the manager of operations. The use of consultant services minimized the political problem, at least during the planning phases. However, the impact or trauma is felt most during the pilot application-system conversion process. The lack of adequate management support to the DBA function as demonstrated by its placement within the data processing organization and the limited personnel resources only aggravated the political problem. As such, the establishment of the DBA function was less than effective. The influence of such political struggles on the suspension of further data base conversion activities is difficult to determine.

A major problem in converting from a nondata base environment to data base is that during the period when data base expertise is critically needed, you are least able to provide it. This problem was eliminated with the expertise provided by consultants.

The placement of the DBA within the data processing organization and the limitation of its responsibilities contributed to the lack of support at the right level of management for continuation of the data base project. This was very much in evidence when staff meetings were conducted to set priorities for data processing activities.

The benefits realized by the conversion and implementation of the pilot application system resulted from the redesign activities. However, the cost/benefit analysis was instrumental in identifying a pilot system that had low risks, early benefits and visibility. These three factors alone may very well save the future of the data base project in the context of further implementations of data base. Once a data base conversion effort is suspended, it normally takes support from the user community to reactivate it.

Problems encountered during the conversion process were due, for the most part, to inadequate training of peo-

ple in the usage of the DBMS and in the interpretation of its diagnostics. The amount of training planned was reduced in an effort to compensate for an insufficient number of people made available to convert the pilot system.

The second conversion experience involved choosing and using a DBMS to support processing of solar hardware system data. The solar hardware data resided in file format, yet structural relationships existing between and among data records could not be supported in any reasonable manner in the file-oriented environment. In addition, many of the user requirements were not adequately defined, and there was considerable worry about whether the known requirements could ever be met.

A group was gathered to define the users' information-processing requirements and to determine whether a DBMS solution could be provided within a reasonable time at reasonable costs. Initially, all combinations of hardware and software solutions were considered because the data processing center within the organization did not have a DBMS. As time progressed, the users' requirements became clearer and time-sharing service solutions—the desired mechanism for achieving a hardware change—were dismissed because of control and cost factors. The conclusion was to provide a DBMS for the existing hardware.

Although unanswered questions remained regarding the feasibility of this approach, many desirable aspects were apparent. The DBMS selected was available in a bundled form from the hardware vendor—no additional costs—and the software could be delivered in a timely manner. In additional, the selected DBMS supported the required structural relationships with its CODASYL orientation.

The best understanding of what output requirements the DBMS should provide was used to design the data base and implement application programs. All of the requirements were met in a reasonable time within reasonable costs. Critical points in the success of this DBMS conversion experience were as follows:

- DBMS expertise was available from the beginning—a most important time in any DBMS environment.
- The users participated in defining the information-processing requirements.
- Expertise was available in matching information-processing requirements to available DBMS capabilities.
- The vendor provided adequate training and sufficient time to gain experience after the training was completed. This provided the necessary insights into the actual capabilities of the chosen DBMS.
- Portions of the output requirements were sufficiently well-documented and were of reasonable magnitude to demonstrate successful operation of the DBMS demonstration.

In the third conversion experience, the following goals were defined:

- To convert a large number of antiquated files to state-of-the-art technology.
- To provide for a more efficient way of retrieving and updating data.
- To eliminate redundancy through a centralized data bank.
- To eliminate erroneous information through data integrity.
- To provide better management of information.
- To provide for more system security.

Only one choice was available to convert existing files to a DBMS and at the time only one DBMS could meet the requirements set by upper management. A hierarchically structured-data model DBMS used exclusively with its own higher-level language interface was chosen.

The first conversion had to execute the load procedures eight times until all known errors were eliminated. Each load took more than 100 hours to run with no checkpoints. Fortunately no crashes occurred during any of the runs. The load process in itself was not a major problem; however, some erroneous data present in the old file was input to data bases. The users were encouraged to be involved in checking and rechecking the new system. At times, the users requested changes which, to them, seemed minute, but which required large programming and system changes. Some users wanted to have more information in the data base (such as segment's last date of change) while others were concerned with space and efficiency. We provided a backup system which we could fall back to in case the conversion failed. Meetings with users continued for six months until repeated testings assured us that we had reached a satisfactory level of accuracy.

Another problem resulted from personnel assignments. While one group was loading the data base, another was testing. The group that loaded the data base did not have as good an understanding of content as the testing group. Consequently erroneous data was loaded. If both groups had worked more closely, most of these problems would have been resolved.

The following points are characteristic of successful conversions in the described scenario:

- Management commitment insures sufficient resources and end-user support of the conversion effort.
- DBMS expertise must be available from the beginning of the conversion effort.
- Adequate planning activities are instrumental in the conversion to data base.
- The end-users must participate in defining information-processing requirements.
- The DBA support must be a strong and continuing commitment.
- In the DBMS package evaluation, expertise must be available to match the information-processing require-

ments to the capabilities of available DBMS packages.

- Adequate and timely training must be provided and sufficient hands-on experience must be gained prior to any conversion of application software.
- A conservative staging plan, one which selects an application system for initial implementation that offers low risks, high benefits and visibility, must be developed to insure continuing resource support.
- The DBMS conversion must be thoroughly and adequately tested prior to the production cycle. A fall-back procedure should be established in the event that the conversion is not successful.

The next two examples are conversions in which data is maintained and used manually in the source (old) environment and it is desired to convert the data and functions directly to a DBMS without first introducing a non-DBMS file system.

The goal of the first example was to mechanize functions performed manually by separate operating departments using the same input documents. The input documents triggered each of these departments to perform its own functions, using its individual, manually maintained data records. The flow of the input documents went from one department to another. The flow was sequential, since one department had to complete its function before the next could begin. Each department independently maintained its own data records. The records in each department contained some common data components.

The total process had never been mechanized before because of the complexity of the operations involved and the lack of data structures to represent the interrelations of the data and functions. DBMS technology made mechanized processing and a common data storage place feasible. The expections of mechanization included:

1. Better performance of end-user functions through more accurate records and increased capability of machine over human operation.
2. More efficient performance of functions through elimination of duplicate record keeping and mechanization.

The project selected a network DBMS including full communications and transaction management capability. Interactive terminals were designated for end-user locations as well as an interface developed with a front-end system controlling the flow of the inputs to the system and the distribution of the outputs to the appropriate user departments and to other mechanized systems.

The conversion from a manual environment to the full DBMS has been completed successfully at one site, and the DBMS has been operational for several years. During this time, more sites have been converted to the system. Interfaces between this system and other DBMS systems are being developed. Since these interfaces are not yet operational, conclusive findings cannot be stated.

The task of interfacing two DBMS should not be underestimated, especially if the systems use different software or hardware. For example, plenty of time should be allocated in developing the interface simply to work out the kinks of reading data created by another DBMS in a language with a different bit structure and character set.

When an interface is first developed between two systems, time must also be allotted to resolving differences in how one system identifies or names what are thought to be common data elements. Both systems may process the same widgets, but one may be concerned with inventory and the other with maintenance or repair. Both systems can be involved with some of the same widgets, but the way they identify or represent them may be different. These differences may not be apparent or may seem trivial until an actual mechanized interface is attempted.

Merely because there could be a mechanized link between two systems does not mean there should be such a link. It may be far more economical to use a manual interface, especially for low-volume interactions or in cases where the viewpoints of the two systems differ greatly.

Environments with multiple DBMS are very frustrating to users if these DBMS each use different input devices with different sign-on procedures and modes of interaction.

Coordination and planning between the DBMS in development are needed to prevent proliferation of different terminal equipment at the same user locations.

In the second conversion experience, the goal of the project was to automate a manual document-retrieval system. Management greatly simplified the decision process by directing conversion to a custom-built document-retrieval system which would be converted from one hardware manufacturer to another. Apparently, the reason for this decision was that a copy of the software could be obtained free.

A software conversion (COBOL-to-COBOL) had to be done before the local document-retrieval application could be automated with the software. Then the software had to be augmented (new code) to add needed functions, including more extensive editing of input data, more user-oriented processing of retrieval requests and redesign of outputs.

After the software conversion was started, analysis made it apparent that the capture of the manual data in a machine-readable form would require about 12 staff/years. Also, on-going entry of new documents and servicing users would have required a permanent staff of four to five people. Attempts to convince management of these needs were unsuccessful.

Subsequently, a successful software conversion was followed by augmentation of the converted software. Then came design of procedures and entry forms and identification of resources needed to capture the manual files. The

project was finally shelved because resources were not available to contract either the data capture or to staff it in house.

While not a conversion to a true DBMS, this experience illustrates, by their absence, several points necessary for DBMS conversions:

- Importance of planning and analysis before any decisions are "cast in concrete."
- In this case, the software was decided upon before analysis. And the software chosen was totally inappropriate for these reasons: It was not running on the same brand of hardware as the local system, so it had to be converted. It was custom-built and no support was available from the originator. Software documentation was fragmentary and obsolete—we got source code, compiler listings, test data and very little else that was useful. The software did not provide adequate functions, so it had to be augmented with new code.
- It is important to obtain a commitment of resources sufficient for the entire project before starting any part of it. In this case, the decision-makers had no comprehension of what would be involved. Neither the one-time data capture nor the on-going support staff needed could be had when the time came to actually implement the retrieval system and service customers with it. Therefore, extensive conversion and analysis were wasted.

The manual-to-DBMS conversion situation has two characteristics which make it unique among the data base conversion situations: data not already machine-readable and users not accustomed to automated systems. Consequently, those contemplating such a conversion should take the following actions:

- It is necessary to limit the scope of the conversion to something manageable and do-able. Limit objectives to high-volume, high-usage functions. Do not try to automate low-usage and/or exception cases. Stick to your initial limited scope. Do not be tempted to agree to ad hoc extensions of scope. Prioritize and save good ideas for future enhancements.
- Plan on phased implementation. Get something useful up soon. (Your limited scope—above—should have selected a useful small application.) Getting something into actual use soon will let the customer see benefits early and provide experience on the system. It also gives the project team some feedback early and buys the project team some credibility with users for later, larger projects.
- Plan on reimplementing. The first limited-scope application(s) will very likely benefit from later reimplementation.
- Select the initial installation/site carefully. Give yourself the best chance of success. If the system is planned for several sites, do the easiest one first.

Understand the magnitude of data capture and data cleaning effort. Everyone, including top management, must understand that capturing the data and correcting it will be a large effort. This effort is a significant part of overall project costs; is may be the largest. It should be understood and budgeted for at the beginning. Sampling the manual data for accuracy and completeness is useful in estimating the resources needed for data capture and correction.

Resolve political problems of ownership of data. Where the data is kept by several organizational divisions, determine which one is the best source for the application you are converting. Designate someone to be responsible for resolving discrepancies such as missing data elements incorrect data elements and obsolete data elements.

These, of course, are familiar functions of the DBA. Such problems are, however, magnified in a situation where there has been no automation before. Early end-user involvement can get some support for the project team and give end-users time to identify changes to their work flow, changes to staff requirements and training requirements.

In going from a nonautomated to an automated environment, higher incidences of errors or data discrepancies are likely to be found. This is especially true in initial installation because the new application software or system software may process the data incorrectly. The system should be designed to be defensive in all aspects—from program design to backup and recovery procedures. The system should be able to handle data discrepancies and inconsistencies gracefully. It should provide meaningful outputs in the face of data conditions it cannot process. It should provide easy to use tools to correct data problems that are found.

In converting a batch file system to a DBMS, the first example involved a federal government agency. The goals of the agency were to replace outmoded hardware, to implement centralized data management under a data base administrator and to implement both new applications and redesigned systems in an integrated data base, online disk and telecommunications environment under a DBMS.

The purpose was to permit simple query capabilities and to treat data so that multiple-use data was input only once, yet have it accessible locally and from distant cities by terminal and allow unstructured queries to be processed through the use of a query language without the necessity for programming

The options were largely dictated by federal procurement regulations. If there were no regulatory constraints, the alternatives would have been:

- To determine the features needed by the DBMS, evaluate existing DBMS and buy a computer on which the most suitable one for their needs would run. This option

was not allowed by the regulations because it would have resulted in a sole-source procurement.

- To include in the request for proposal and DBMS required features. This option was not allowed because, in their case, the requirements would have unduly restricted competition.
- To procure the hardware under open competition and procure the software separately.

However, regulations do exist, and the agency had to select the final option which the procurement regulations had forced upon them.

The results, in terms of their goal of creating an integrated data base under a DBMS, were disastrous. The computer system that was procured was one for which a suitable DBMS did not exist. After two and one-half years of effort to procure, through established DBMS software vendors, a system which could be made to function on this computer, either through conversion of an existing DBMS or the use of research and development software, the agency has not yet found a solution.

Even after a contract is eventually awarded, the agency anticipates a 12- to 18-month wait for delivery of the product. Therefore, it is safe to say that the goal of moving onto new hardware and a DBMS has been so delayed by federal procurement policy that it will take three-and-a-half to four years for the agency to recover.

In the second conversion experience of a federal government agency, the goal was to convert three stand-alone systems with the following features: batch system inputs manually transcribed from messages onto cards and outputs to other subsystems in the form of card images on tape that was hand-carried, as well as printed reports.

The new system was to be online with all error corrections processed and corrected interactively on a CRT terminal. Outputs were to include printed reports with ad hoc queries available through a DBMS query language. Outputs to other systems were to be automatically generated and forwarded via a communications network.

Conversion is usually a one-for-one affair. Redesign occurs when the requirements are changed. Conversion is usually an excuse for partial or complete redesign. A common mistake is to assume that for the price of a conversion effort, one can also redesign. Experience has shown that this is not the case. Unrealistic estimates of needed time and resources result.

The need for expertise in the new DBMS is greater at the beginning of a project than at any other time. Ironically, this is the time before your staff is retrained and when your expertise level is the lowest. Therefore, outside consultants must be brought in. These outside experts will have a great impact on your data base design.

The DBMS experts will, at an early stage, have to design the data base structure. The system design will then build upon this data base structure. The process of data base design and system design will be iterative with a series of changes until both are in sync.

The extreme pressure to get off old machines and operational on the new machines prevents sufficient time for thorough analysis and planning. And, as discussed above, if you are redesigning and not converting, the time pressure becomes greater.

Government procurement regulations do not recognize that a system is as dependent on the software needed as it is on the hardware.

The move from separate batch systems to an online integrated system, coupled with the need to learn how to use new hardware, require careful planning and extensive training in both new concepts and new techniques.

Moving into the centrally controlled DBA environment requires lead time in the systems development cycle for data dictionary development and implementation and for data standardization to be done carefully.

Time required for learning new concepts and techniques is seldom available when a move to new hardware is under way. The result is hasty and costly straight conversions of obsolete systems; therefore, the potential benefits of the new hardware are delayed or lost.

The next conversion experience involved moving from one DBMS to another, with a change in vendors. The goals were as follows:

- To provide for the ability to update concurrently and retrieve information from the data base.
- To provide for a quick response time for inquiry purposes.
- To provide for the capability to have online inquiry even if the DBMS is in abort status or in recovery stages.
- To provide an efficient way to recover and back up the data base.
- To save money in the long run.
- To provide tools for security and integrity of the DBMS.

At the outset, three choices were considered. One was to continue with the present DBMS. This implied maintaining duplicate data bases—one for inquiry purposes and one for updating purposes. The inquiry data base was at least one day behind the second one and at most a week behind. To make the inquiry data base as current as possible, an image copy was made once each week of the updated base and copied to the inquiry data base.

A second choice was to update the DBMS using the same vendor. The third was to convert to another DBMS using a different vendor. The new vendor agreed to convert the bulk of the system quickly and with the least amount of logic changes to existing modules. This vendor would meet most if not all of the goals.

Upper management decided upon the third course. To meet the restriction of making no logic changes, the new vendor developed an emulator to translate dynamically (at execution time) calls in COBOL programs using the old vendor format without requiring manual recoding. Only format changes were required, and these were handled automatically via a standard vendor software package.

The emulator itself was developed by some of the best technical people available at the vendor site. It sounded and looked great—minimal programming changes and transparent to users. In fact, the programs looked as if they were written in the old vendor DBMS. The emulator did, however, have certain drawbacks. The overhead incident to using the emulator was extremely high, since the calls were converted at execution time rather than at the source level.

Using the initial version of the emulator would have resulted in failure to maintain the necessary production schedule. A task force was brought together to assess the matter and determine exactly how and where the system could be improved. The task force included site and vendor staff.

The task force recommended changing some emulated programs to execute under standard software and looked into the emulator internals in an attempt to improve some of its functions. The changes yielded significant improvements; after the changes were effected, the regular production schedule was met.

Another significant problem arose when the emulator did not perform properly, mishandling a call to the data base. On the rare occasions when a malfunction of this sort occurred, the following steps were taken to solve the problem: disable the erroneous call; inform the vendor of the problem, providing complete documentation on the problem; wait for the solution, retest the call and reload the new version of the emulator to the system library.

The prospect of losing technical support for the emulator, in the event of personnel turnover among the vendor staff who designed and developed the system was another potential problem. This situation did materialize, for the vendor was able to continue technical support with other individuals.

Conversion of the data bases themselves went extremely well. Standard utilities were used to unload the data bases. The tapes were moved physically to the new vendor's facilities and were converted via a standard vendor utility. The subfiles were loaded employing user-written programs.

Since the online system had been installed on the new system well before the other applications systems, there was a need to insure that the data base on the new vendor system was as current as the old. With smaller data bases, we unloaded daily from the old vendor and loaded daily on the new. With the exception of our largest data base, the other data bases were unloaded and loaded weekly or monthly. To keep the largest subfile current on the new system, we captured the data base changes on the old system daily and applied them to the new data base. These procedures lasted for approximately one year until our other applications systems were completely converted.

To summarize our findings:

- Differences between terminology associated with each system were found to be the initial barrier.
- Incremental transition was recommended as a means of graduated phase-in to the new DBMS.
- An intermediate software system was procured as the method of changing application code. DDL and DML functions appeared the same across the transition. Run-time interpretation of code through calls to interpretive software was used to provide a mapping from original DBMS to native-made DBMS calls. Preliminary conversion of DDL-type facilities was provided at compile time with mapping structures being derived for use at interpretation time.

Experience in interpretation and run-time mapping has shown that system overhead can be prohibitive. Problems encountered in execution require resolution by the developer of the interpretive software and can result in excessive response times. It is thereby seen to be more practical to utilize standard vendor software where possible. Conversion costs may be higher in the initial investment, but will eventually be equaled and even surpassed by the overhead costs of the interpretive method.

The development of the interpretive software was handled by the vendor, with valuable insight of users. User involvement in the implementation of the interpretive software provided a more timely product with greater response to requirememts.

The interpretive software, by nature, required processing support over and above that required for actual application processing. Direct transition could possibly have utilized to a greater degree the processing efficiency of the target DBMS. Again, this leads to the realization that should efficiency be a major consideration, direct transition may be more desirable.

In utilizing direct transition methods, vendors should be encouraged to develop standard methodologies for conversion processes in data description, data manipulation and data base transfer.

Incremental transition insures that timely conversion can be established for applications. Prototyping efforts are provided through validation of earlier increments of the entire transition process.

Fall-back positions must be established. The transitional software must be completely validated before actual acceptance. The ability must always exist to turn down changed software due to errors without degrading overall effectiveness of the data processing operations.