IMS: PAST, PRESENT, FUTURE

Born of an affair between IBM and Rockwell, it put men on the moon and billions of corporate records on-line.

by William P. Grafton

Fifteen years ago, in the early morning hours of Aug. 14, 1968, a group of excited people at the Rockwell Space Division plant in Downey, Calif., watched as the words "IMS READY" printed on the 2740 typewriter Master Terminal, and the operator entered the command, "/START REGION 0." IMS/360 was about to go into commercial production for the first time. Almost immediately, a stream of transactions began arriving from the manufacturing shop floor, as workers entered queries and status about work orders for the Apollo Project. Two hours later, I turned to Hugh Hoskins and said, "I think it is going to stay up. Let's go get some breakfast." As I recall the moment, Hugh said, "You go. I think I'll stay here and watch it a while longer."

These thoughts about the history of IMS are based largely upon material prepared for a presentation I gave to IBM at Santa Teresa in May 1982. At that time, I made the comment that I felt as if I had been invited to give a lecture on Christianity at the Vatican. Nevertheless, if IBM Santa Teresa is the Rome of IMS, then perhaps I can

If IBM Santa Teresa is the Rome of IMS, then perhaps I can say I was present at its Bethlehem.

say that I was present at its Bethlehem.

That, of course, would be the Space Division of North American Aviation (Rockwell International), where IMS was born. While there, I participated in the early development of IMS and had the privilege of managing the first production IMS installation.

I have been continuously involved with IMS since that time, in hardware and software management, in networking and distributed processing, in application development, and in data and database administration. During the 15 years that IBM has spent developing and marketing IMS, I have been busy trying to make it meet the information needs of business. This activity has included employment at three major IMS user companies, consulting assistance to five other large IMS installations, membership in several IMS user organizations, and technical presentations to a number of educational, professional, and technical institutions.

As many know, IMS is the illegitimate offspring of an affair between IBM and Rockwell International. In 1961, Rockwell was selected prime contractor of Apollo, the largest single engineering undertaking ever contemplated. The need was recognized for mechanized control of the engineering data involved. A special requirement was an automated indentured parts list that would associate all of the parts necessary to manufacture a complex end item. There grew to be about two million parts in the Apollo spacecraft.

HISTORY OF EARLY SYSTEMS

There was no technology at the time that satisfied the requirements, so a magnetic tape-based system was developed, incorporating a complex search technique that used core storage as a pseudo-direct access device. The system worked but was extremely inefficient. The file occupied 18 reels of tape, with low activity against any specific record. Sixty percent of the file was redundant repetition of assembly and part numbers, next items, effectivity, etc.; machine time was excessive; and the batch processing technique meant that the file was never up-to-date.

It was determined that the next step should be a generalized file access method that was direct-access based. The method had to be one that could be taught quickly to programmers with little or no direct access experience. It had to be capable of processing hierarchical file structures such that file management techniques eliminating redundant data could be employed, and it also had to be relatively device and language independent.

The resulting software was called GUAM—Generalized Update Acess Method—and was the forerunner of Data Language/One (DL/I). It was used to implement the Disk Oriented Engineering System (DOES) at Space Division in September 1965, utilizing the IBM 7010 and 1301.

Rockwell developed two Apollo teleprocessing applications in parallel with DOES: the Engineering Document Information Collection Task (EDICT) and the Logistics Inventory Management System (LIMS).

EDICT was designed to track the current status of engineering drawings and specifications. The Apollo effort was worldwide, and a request for status information could originate almost anywhere. The IBM 1460 was the central processor for EDICT, which utilized the 7770 Audio Response unit and 1301 and 1311 disk storage. A series of 1026 control units monitored and controlled input from twenty 1050 terminals. LIMS used essentially the same configuration, with the exception of Audio Response, and allowed on-line update and inquiry about the status of critical parts in the Apollo project.

The teleprocessing monitor that supported EDICT and LIMS was known as RATS—Remote Access Terminal System—and was developed jointly by Rockwell International and IBM during 1964-'65. It was a generalized system that performed the functions of polling terminals, interpreting messages, and calling the application programs. One message at a time was processed with no task switching interrupts. It was the forerunner of IMS DC.

A new type of redundancy was now recognized. Half the data in a DOES record were identical to that already existing in an EDICT record, and 99% of the EDICT records were also in the DOES file. Combining the two files, however, would require recoding both systems. This was the situation when System/360 arrived. It was decided to exploit the capabilities of the new computing system by designing a software package combining the best features of GUAM and RATS, and adding capabilities for concurrent message processing; external definition of file structures; protection of sensitive data; improved search, retrieval, and storage techniques; multiple device support; and other features. IMS was conceived.

THE IBM ROCKWELL PROJECT

Dr. Robert R. Brown, director of data processing at Rockwell, formed a joint project with IBM to develop the new package. The product was initially called ICS (Information Control System) but was later rechristened IMS when IBM uncovered some sort of trademark or copyright problems with the original name. Dr. Uri Berman of IBM and Bob Patrick, a senior consultant, developed much of the original architecture and specifications. Ed Morris of IBM was named project manager. Pete Hill of IBM and Pete Nordyke of Rockwell were named co-development managers. Pete Hill assumed the project management role on Jan. 1, 1968, and led the project during the crucial implementation and product development years. Some of the key development personnel were: for DL/I—Dan Gilbert, Pete Nordyke, Marv Nichols (Rockwell), Uri Berman, Sid Kornelis (IBM); OSAM—Lee Meador (Rockwell); scheduler—Don Lundberg, Thomas Work (IBM); buffer management—Tom Sawyer (IBM); system macros—Craig Franklin (Rockwell); checkpoint/restart—Don Hyde (IBM), Earl Carbone, Hugh Hoskins (Rockwell); teleprocessing—Les Premo (Rockwell), Carl Chamberlain, Howard Keller (IBM); audio response support—Bill Erwin (IBM); and for documentation—John Calvert (IBM).

The bulk of the system design work was completed during 1966-'67, with coding and checkout taking place in 1967-'68. The development machine was a 512K S360/50. The work was done at the Downey, Calif., facility of Rockwell Space Division.

In parallel with the development of IMS, Rockwell was conducting beta test implementation of OS/MVT in order to have an operating system that could support the multiple control regions required by IMS DC. During 1966-'67, I was helping to develop the controls, procedures, and operational environment required to run OS/360, and to drive the conversion of over 100 applications from 7010 to S/360 technology.

My association with IMS began in the spring of 1967. Bob Brown was scheduled to give a talk on "ICS" at the International Federation of Information Processing Societies (IFIPS) conference in Rome, and asked me to help him prepare the speech. The research resulted in a document that for the first time presented a comprehensive overview of the objectives, philosophy, architecture, and structural organization of IMS. Bob was pleased and the speech was a great success. Not long after his return from Rome, Dr. Brown transferred me to the project team.

My "surface" assignment was interesting. I was to work on the man-machine interface to IMS—the terminal command, the master terminal function, the operations procedures, the manuals, the training and education. But, there was a second and even more intriguing covert assignment. Dr. Brown was extremely concerned that the IMS Project was falling seriously behind schedule and might well be out of control. Many Apollo flights were upon us, and the first lunar landing was only a year and a half away. It turned out that my real assignment was to determine the status of the project and to recommend the specific actions required to implement IMS as a production system.

In March 1968 things were beginning to turn around. Then the Rockwell development manager and most of the Rockwell IMS team resigned.

What I discovered and reported to Brown was not encouraging: there was no detailed implementation plan that anyone was seriously attempting to manage. The Rockwell team had become alienated from its management, and important information about status and project activity was being withheld. The project was indeed out of control; there seemed to be little sense of urgency or personal commitment to the Apollo Project among team members.

The Rockwell team was expending a great deal of effort on IMS enhancements and extensions, such as an on-line query language, that were outside the scope of their mission. This was at the expense of completing the basic product. The IBM team was busy redesigning and recording functions that had been reported complete months earlier. I was not to understand the motivation for this until the unbundling announcement sometime later in 1968.

There was no comprehensive testing program to exercise the system methodically, identify problems, and fix them. IMS simply would not run reliably and no one was doing anything about it. The project team had developed an elitist "priesthood" attitude toward the application development group, who were trying to implement three major on-line systems under IMS. The project team hogged the computer resources, crashed the system repeatedly, ruined application tests, and destroyed databases. Application programmers who were seeking help were treated with disdain.

NEED FOR DRASTIC ACTION

Brown asked me what I thought he should do about the situation. I recommended drastic action:

- The joint develoment relationship with IBM should be terminated. The mutual interests of the two companies had diverged. IBM wanted to develop a marketable product. Rockwell wanted to go to the moon.
- The present design of IMS should be frozen and the Rockwell team should concentrate its efforts on making it work.
- The IMS development machine should become an implementation machine. IBM should move its development efforts elsewhere.
- Considerably more attention should be paid to project management and control, testing, application support, and the operational environment.
- A few prima donnas should have their attitudes adjusted.

Dr. Brown then asked me when I thought IMS could be ready for production should he follow my advice. I told him by July 15, 1968, one year prior to the planned landing on the moon.

I am sure that I was not the primary catalyst for the events that followed. Others must have observed the facts that seemed so obvious to me. Dr. Brown himself must have had a pretty clear idea, or he would not have asked me to investigate. Nevertheless, the joint project was ended, IBM moved the development team to Century City, and the design for the first implementation of IMS was frozen. The Rockwell team was directed to concentrate solely on implementation.

I set out to test the system command by command, module by module, transaction by transaction, function by function, utility by utility. Every time I found a problem, I gave it a number. I organized a problem resolution committee that met almost daily to classify the problems, determine priorities, and assign responsibilities for solution. IBM was a member of the committee and was given a copy of every problem. We kept in close telephone contact with IBM team members in Century City.

I also developed a PERT schedule of major implementation events and activities in order to track progress. Jo Ann Storts and I put together a master terminal room, trained the first IMS Master Terminal Operators, and wrote an MTO handbook. I blocked off a corner of the machine room and reorganized the transmission controllers, modems, dial sets, and plug boards into an embryonic network control center complete with a secondary master terminal.

In March 1968, things were beginning to turn around. Then, the Rockwell development manager and most of the Rockwell IMS development team suddenly resigned from the company en masse. Dr. Brown asked me if I thought he should write off the project and give Apollo management the bad news. I told him no—there was a nucleus of good, dedicated people left, and it could be done. Brown asked when; I said push the date ahead a month to Aug. 15, 1968. I walked out of his office as manager of the IMS project. During the next four months we cleared up over 200 system problems and completely rewrote the database recovery facility. In mid-August, the system went into production on schedule on a S/360 65 with 512K bytes of memory. It has been running ever since.

Gene Brault and Hank Epstein managed the first group of IMS applications with supervisory support from Al Barnett, Bob Whitaker, and Dick Duffy. Jim Lightfoot and Ed Duncan were the development project leaders. Some of the key programmers involved were Rod Shahanian, Dan Weller, Dave Johnson, Carol Roark, George Foote, and Roy Gray. Implementation was performed on a step-by-step basis. Complexity was added gradually. During 1968–'69, we implemented eight applications. The first IMS application in August 1968 was POLAR, a Production Order Location and Reporting System that featured uncomplicated databases, 2740 terminals, and simple transactions.

For the statistics buffs, by 1969 the system utilized 130 terminals and 110 lines; occupied four 2314 units for 30 databases spread over 32 disk packs; generated 17,000 to 20,000 transactions a day; supported 260 transaction codes; operated on a 20-hour day; and had an average response time of two to five seconds.

FEATURES OF IMS DESIGN

For marketing reasons, IBM insisted that IMS be able to run on a 256K machine. This restriction permeated the IMS design, affecting everything from what functions would be implemented to module sizes, queueing strategy, control block limitations, and programming techniques. If this seems odd in these days of multimegabyte memories, consider that when IMS was designed, memory technology was magnetic core based and very expensive. There was no virtual storage, and 256K was a reasonably large machine.

IMS was built on top of OS/360 as an extension of, but not a part of, the operating system. I believe this was done because IMS was developed as a Type II Program by the Manufacturing Industry Development Group of IBM, while OS/360 was a Type I Program out of that holy of holies, the Data Processing Division. Navigating the OS/360 interfaces was probably less traumatic than getting two different IBM organizations to cooperate with each other. In addition, OS was every bit as new and untried as IMS; and the development team members probably felt they had enough variables to deal with without having to cope with integrating IMS into OS/360. For whatever reasons, IMS was layered on top of OS, and there it sits today—passing, posting, queueing, saving, restoring, interrupting, masking, and boundary crossing.

Why were hierarchical databases chosen for DL/I? I can remember the debate at Rockwell. There were advocates of the network approach being used by Bachman at GE, and of the inverted file concept used by some of the library automation projects.

But disk files were small at the time, and the Apollo storage requirements were large. Hierarchical storage techniques conserved disk space. Rockwell and Caterpillar had an urgent need for parts-list and bill-of-material processing, which were natural hierarchical database applications. Finally, the GUAM software mentioned earlier was the forerunner to DL/I, and it was based on the hierarchical model.

For relinquishing its rights to IMS, Rockwell received an acknowledgment, a waiver of license fees, and 10 free sets of manuals.

There were strong convictions among many of the project team, myself included, that IMS should be driven by an integrated data dictionary—that all data entities should be defined and all data accesses controlled through a common facility. The proposal was defeated, a victim of schedule pressure and the 256K limitation. I am sorry we lost that one!

QUESTIONS OF INTEGRITY

One subject that found universal acceptance throughout the development team was the principle that data entrusted to IMS should not be lost, corrupted, or compromised; and that the system should be immune to bad data, bad programs, and bad operators. I believe that this philosophy stemmed from the extreme safety and integrity requirements of the Apollo program. A few anecdotes may serve to illustrate the point:

Automatic backout of aborted transactions. In our testing of an early version of IMS, we demonstrated that it was likely that an abend of IMS or an application program would leave a database in damaged condition. In this case, a full forward recovery was necessary before restart could be attempted. This was clearly unacceptable from a user service standpoint, but it was the way the system was implemented.

Don Hyde of IBM did not like the situation. He proposed a revision of the checkpoint/restart architecture to include automatic backout of partially completed transactions, and provisions for rescheduling them during restart.

This sounded like a major effort to me and I said so. Don assured me it was "no problem." Such statements tend to terrify me, but Don was as good as his word and had the modifications coded in an amazingly short time. The changed system sailed through regression testing without major difficulty, and we now had much better database intergrity. I believe this improvement may have been the most significant factor in making IMS an operationally viable system—and I almost vetoed it for the intitial implementation!

Improved database recovery. Our testing of IMS utilities showed conclusively that the first version of database recovery was not reliable. It was based upon the concept of restoring the database from the last unload tape, and then reprocessing all subsequent transactions against the database up to the point of the failure.

Marv Nichols and I developed a new database recovery method. Don Hyde had written code to record all database update "before" images on the IMS log in order to affect his backout and restart capability. Marv and I extended Don's code to record the "after" images also. Our recovery technique merged the database unload tape with subsequent after images from the IMS log in a single batch pass that produced a recovered, reorganized database. This assured an accurate recovery, reduced the time for recovery by an order of magnitude, did not reprocess transactions, and did not require IMS to be up. IBM later adopted a similar approach to recovery in IMS/360, Version 2.

Quality assurance testing. Our techniques for system acceptance testing proved invaluable in keeping bad code out of the system. We developed a battery of test scripts and cases, test data, and special testing utilities. Whenever a bug slipped by us, we installed a test in our arsenal that would have caught it. We adopted the position that nothing that IBM gave us was any good until we had tested it and proved otherwise. Whenever we uncovered flagrant examples of destructive or unexecutable code in delivered software, we blistered IBM and demanded that they do a better job of testing their work before release.

At this point, I want to emphasize that I have the highest regard for IBM, both as an organization and as a group of extraordinarily talented and dedicated people. Many of the IBMers with whom I worked on IMS have become lifelong personal friends. I have never, before or since, encountered a team that gathered together in one place so much talent, integrity, and fellowship as the IMS Development Project. They were the best.

The problem was that IMS was the first, or nearly the first, large commercial program product ever marketed by IBM. Added to this was the fact that IMS represented a new way of doing business to its users, and the customers were betting their companies on the reliability and availability of the IMS DB/DC system. I do not believe that the associated product quality implications were fully understood in the beginning. Eventually IBM created a quality assurance organization for IMS that adopted much of our philosophy and methods. Soon, IMS became one of the most solid software products available, with a well-earned reputation for reliability and integrity.

WHY IMS WAS A SUCCESS

I estimate that IMS must generate at least $50 million in revenue per year in lease and license fees. When the income from associated sales and leases of supporting software products, terminals and controllers, modems and communications processors, direct access storage, and large mainframes is added, one must conclude that IMS has been one of the most successful of all program products. It would be useful to know why.

Success was obviously not self-evident from the beginning, at least to some folks. When Rockwell negotiated the termination of the Joint IMS Development Project with IBM, Rockwell relinquished its rights to the product in return for: 1. an acknowledgment on the inside front cover of the first issue of the manuals, and 2. a waiver of license fees, and 10 free sets of manuals for the first three releases of IMS. Those of us on the Rockwell team considered that IBM had struck the greatest bargain since the Dutch bought Manhattan from the Indians.

These are some of the most important factors in the success of IMS:

IMS works. The flexibility and power of DL/I have been used to solve the database problems of the world's largest and most complex organizations. Its data integrity protection is so reliable that these companies have entrusted it with their primary financial, marketing, product, and personnel records. The IMS data communications architecture has the capacity and operational reliability to put an entire enterprise on-line, with assured growth potential for the future.

S/360 compatibility. Its compatibility with S/360 and OS/360 was a key factor in the success of IMS.

SHARE/GUIDE contributions. The SHARE IMS Project is discussed in detail later in this paper, because I was personally involved with it, but I certainly do not mean to diminish the importance or contribution of GUIDE.

Project management. The engineering project management approach to development resulted in a product that was a true system, was technically sound, and was operationally reliable.

End-user involvement. The bundled environment in which IMS was developed fostered a free and open exchange between developer and user that is lacking when development takes place in an ivory tower atmosphere. IMS was designed and built on-site by the end-user and industry specialists from IBM. It filled a critical market need at exactly the right time.

Vendor support. IBM product support and the commitment to continuous enhancement, along with upward compatibility, built customer confidence in IMS as a long-range product direction.

Integrity and recovery features. The data and system integrity and recovery features of IMS were superior to competing products.

Pete Hill. The charsima, leadership, energy, and commitment of Pete Hill were of incalculable value in the success of IMS.

In 1969, I was asked by IBM to attend the SHARE summer session in Boston, to discuss the possibility of organizing a joint

The truth of the matter at that time was that IMS wouldn't work as received from IBM.

SHARE/GUIDE group made up of the beta test users of IMS. The initial meeting was sponsored by the database committee of the Data Management Project, with Jim Frye of Mitre as chairman. The net result was the formation of an IMS Subcommittee, led by one of the real giants in the saga of IMS, Joan Heinonen of TRW. Her leadership, courage, and sound policies were instrumental in the growth of SHARE/IMS from a subcommittee of six in Boston to its status today as a full division with hundreds of members and dozens of projects, committees, and subcommittees of its own.

The joint SHARE/GUIDE aspects of the organization did not work out because SHARE and GUIDE themselves were attempting to merge at the time, and the effort failed. Instead, each IMS group decided to go its own way, and I elected to stay with SHARE. The founding members of SHARE IMS were: Joan Heinonen; Clifford Pasley, Caterpillar Tractor; Daniel Brooks, LTV; Richard Lewis, First National Bank, Chicago; Ronald McDowell, Chevrolet; and myself.

Joan succeeded in establishing a rather remarkable relationship with IBM. She persuaded the company to sign a nondisclosure agreement with each of the individuals involved. This arrangement facilitated closed-door sessions between the group, who were all IMS beta test participants, and Pete Hill and other IMS development team members. Thus, the tradition was established of direct communication between the users of IMS and its developers. The power and flexibility of IMS today is due in large measure to this communication.

Joan Heinonen established a policy of closed working sessions during the first three days of SHARE week. Open information meetings, round table discussions, and user experience presentations were scheduled for later in the week. The policy of work sessions was fruitful. Many of the eventual external design features of IMS/360 Version 2 and IMS/VS were hammered out at SHARE/IMS meetings and presented to IBM as resolutions. Jerry Kral, of First National Bank of Chicago, led much of this effort. The closed working session technique is now common throughout SHARE and GUIDE.

The closed session concept did not sit well with some of the old-time SHARE attendees who loved to roam the halls of the conference headquarters, wandering in and out of meetings without ever producing anything. One disgruntled attendee who found his way into a closed session blocked by Joan (who was formidable), complained to SHARE management that the IMS Project was a secret society run by a "dragon" who would not let anyone in. This comment became an instant classic. Joan was forever after known as the "Dragon Lady," and the symbol of IMS came to be a huge green dragon straddling the globe.

Another valuable product of the SHARE/IMS Project was the publication of IMS Flyers. These were papers authored by project members and sent to all the membership. Dan Brooks submitted the first flyer. Lew Bethards, of the Federal Reserve Bank, Kansas City, made a major contribution by taking care of all the printing, mailing, and filing work.

After the first issue, the flyers languished. A few more were submitted, but they were mostly lightweight. I decided to do something about the situation. The truth of the matter about IMS at the time was that it wouldn't work as received from IBM. We at Rockwell had performed major surgery on the product in order to implement it as a useful production system. I decided to publish the key results of our work as IMS Flyers, so that other users could get off the ground. The subjects ranged from bug fixes and code modifications to operational procedures, parameter settings, and analysis techniques.

The results were electrifying. Tom Schroeder of United Technologies contributed a group of equally meaty documents, and other members of the project followed suit. The logjam in IMS was broken. A set of the SHARE/IMS Flyers became a required acquisition in every IMS technical library. Without them, I think that the majority of the users would have abandoned IMS.

One of the wisest actions of the Steering Committee was to avoid perpetuating itself in office. In order to give new blood a chance at the enriching experience of managing SHARE/IMS, the founding members eventually founded a "Geriatric Committee" and designated themselves members emeritus. This status permitted them to give advice and counsel and to attend the nondisclosure sessions with IBM, but turned over the leadership of the project to bright new talent such as Tom Schroeder of United Technologies, Hugh Hoskins of Rockwell, Gary Polette of MacAuto, Cathy Stanley of John Deere, Bob Ojala of Motorola, Jerry Kral, and Mike Soulakis of Mellon Bank.

There is one more story that must be told about SHARE/IMS. Joan Heinonen, who could not be outmaneuvered or outfought by any human adversary, fell victim to a crippling spinal problem and had to retire from the computing profession. She is confined to her home in Laguna Hills, Calif., with her body broken but her mind as sharp as ever.

When Joan had to withdraw from SHARE/IMS, the job of leading the project fell to Bill Petefish of Caterpillar Tractor. Where Joan was fire and ice, Bill was calmness and efficiency. He brought a professional management perspective to the organization exactly when it was needed. IMS was no longer a minor product, and the IMS Project was no longer a minor part of SHARE. Bill formalized the relationship with IBM, as the product, the development team, and SHARE/IMS matured. He managed to keep the communication process going while the IMS dragon came to straddle the world, and SHARE/IMS became the biggest division in the SHARE organization.

IMS IN THE PRESENT

One of the earliest promises of IMS was that we would be able to put all corporate data on-line; eliminate redundancy; assure currency, consistency, and accuracy; and deliver management information when and where it was needed. This is the concept of the integrated database environment, where data are considered to be a corporate resource in the same sense as cash, inventory, and receivables.

Has this promise been realized? How successful have we been with the integrated database? The answer I feel forced to face is, not very. Most companies have implemented a few operational on-line systems, wherein the day-to-day transactional activities of the company have been automated. The databases for these systems hold information at the data item level. The problem is that this type of operational data is not very useful for management decision making at the tactical and strategic level. It must be summarized, aggregated, synthesized, and combined with information from other sources in order to be meaningful. It must be compared with historical data so that trends can be determined. It must be projected and extrapolated to explore "what if" situations. Our present database technology is not very good at this sort of thing.

Some companies have implemented tactical on-line support systems for second- and third-level management to control departments, territories, product lines, and so forth, but few, if any, companies use their database to assist upper management in strategic decision making.

Many companies do not even have a true database administration function, other than a technical service to install IMS releases, run DBD and PSB gens, and do database reorganizations and recoveries. Databases are merely on-line files that belong to individual applications and are designed and maintained by application programmers. Little or no attempt is made to coordinate data names and formats or to reduce redundancy. Data dictionaries are rare.

All of the database administrators, data administrators, information resource managers—call the job what you may—that I have met, regardless of whether they have

The pure classic integrated database approach is not feasible with current technology and we should stop kidding ourselves that it is.

been using IMS or another DBMS, are frustrated, discouraged, and disappointed by their lack of success. They are understaffed, underbudgeted, and underappreciated.

WHY SO LITTLE SUCCESS?

College texts, technical journals, the trade press, and the seminar circuit abound with material about the integrated database. A casual observer could be forgiven for assuming that there is frenzied activity in the field. Yet very little of substance seems to be happening. Why? I think I know some of the reasons.

Management's perception is that the bill is too high for what you get. There is a high front-end cost to be eaten, and the benefits are seen as largely intangible. We evangelists of the database concept must do a better job of selling our product as a real financial benefit if we expect to change this. Managers are also put off by the long implementation lead time for the classic approach. They would love to have detailed information at their fingertips. The trouble is that they want it next week, not in five years.

The database approach is also tough to sell politically and organizationally. Plans for developing the integrated database environment require extensive cross-organizational cooperation and commitment of resources. Typically, the database project manager is new, at staff level, has a strange vocabulary, and sounds as if he wants to change overnight everything that the traditional line organizations have been doing comfortably for years. Another problem stems from the technology itself. The available information modeling methodologies and database design tools are inadequate, incomplete, overlapping, and labor intensive. The data dictionary does not support the methodology.

All these difficulties are exacerbated by a lack of enthusiasm in the data processing community. The traditional application development organizations have been slow to adopt the productivity tools that are available and seem content to muddle along with conventional files and COBOL. Databases, application generators, query facilities, and report writers are here, but the community has been slow to adopt them. Even after they do install database technology, many application shops continue to treat databases as though they were on-line tape files.

Finally, a company may not need or want everything on-line. Some application systems may be purchased packages whose data standards are incompatible with those of the master database plan. There may be dozens of applications in existence that work well but do not match the naming conventions or record formats of the database plan. There are probably dozens or hundreds more that should be converted to database, but must await funding and programming staff availability. Meanwhile, the integrated database concept remains a dream.

My feeling is that the pure classic integrated database approach is not feasible with current technology, and we should stop kidding ourselves that it is. There is an alternative approach, however, that will work. I call it the decoupled database concept. Database purists may call it heresy.

DECOUPLED DATABASE CONCEPT

In this concept, the firm is viewed not as a monolith, but rather as a set of decoupled functions that work together: manufacturing, engineering, financial, personnel, marketing, etc. The theory is that each of these functions is a mini-business, and that the information relationships between them tend to be relatively few, straightforward, predictable, and controllable when compared with relationships that exist within a function.

Each of these major business functions is viewed as a family of applications that share a common database. Thus there could be a people database, a money database, a product database, and so on.

The BSP and information modeling processes may now take place at the major business function level according to *functional* needs, policies, and economics. These individual functional information models may then be stitched together as they are completed, thus permitting the firm to converge on the classic *corporate* integrated database model over time. This technique permits incremental implementation of functions, databases, and applications, provides a certain amount of database damage isolation, and allows piecemeal database housekeeping. The resulting "converged" corporate model may be somewhat less pristine than one developed with the classic approach, but it is also much more likely to happen.

Controlled redundancy of data elements in different families should be considered, the better to decouple functional databases from each other. For example, some part number data might be kept in both the product and engineering databases. The control, coordination of multiple updating, and extra storage that this practice entails seems a small price to pay for the development flexibility options it provides. Logical connections between database application families should be kept as loose as performance considerations permit, preferably at the DBMS call level, rather than with DL/I logical or physical relationships. This practice also serves to decouple functions from each other.

Packages should be purchased not as individual applications but as families, with a common IMS database, from a common vendor (e.g., MSA, UCC). There is a caveat here, however. Some of the most popular "IMS database" packages are conversions from batch systems. These systems tend merely to use IMS DC as a terminal monitor, and IMS DB as a disk access method. They are not really adaptable to an integrated database environment, because their data formats have not been put in the IMS database definition but are embedded in the application code. It is difficult to access these databases from other applications (or from a query language or report writer) unless the programs are equipped to handle such a situation, and the vendor includes the formatting scheme with the package documentation.

Applications should be implemented as family units. Interfaces with yet-to-be-converted files should be written as though the interface were actually to a database, by means of the GSAM or SHISAM access methods, and/or a "throwaway" simulator module that traps application program database calls, converts them to whatever data access protocol is needed, and provides the proper database return codes to the program. When the file is converted, the simulator can be removed, and the program can then access the real database without change.

Operational-type applications should be written first. Later, tactical and strategic information may be developed from the operational database and made available to upper management—perhaps in a simple relational or other user-friendly database—for processing with an interactive query language, a report generator, personal computers, or in-house timesharing. This is a very important service an information center can provide.

The mass of old, second-generation tape systems can be given new utility while they await conversion to database. By converting the files to VSAM, and using a full function report facility with an IMS interface, the important tape master files can be loaded into a simple (e.g., SHISAM) database after each batch run for on-line access with an interactive query facility. This simple technique can make a hero out of the database administrator and can hasten the day when the integrated database becomes a reality. ✲

William P. Grafton has an MBA from the University of Southern California and was a member of the IBM-Rockwell-Caterpillar team that developed IMS. He implemented the first IMS production system, managed it for three years, and subsequently became Rockwell's corporate database administrator. He's currently a consultant specializing in database matters and based in Santa Monica, Calif.