EDP ANALYZER
Editor and Publisher: RICHARD G. CANNING

MARCH, 1983
VOL. 21, NO. 3

Our 20th Anniversary Year

REPLACING OLD APPLICATIONS

Many information systems executives feel trapped by the past—they have hundreds or thousands of old application programs and data files that they would love to replace. But with a backlog of perhaps two years' worth of new work already in the queue, they see no way of replacing these 'dinosaur programs.' This need not be the case any longer. Data management systems (and their fourth generation languages), application generators, and application packages now make replacement of many old applications possible and practical. It is time to establish 'a planned replacement program.' (An executive summary is on page 16.)

Irving Trust Company is a major money center bank with headquarters in New York City. It is the 16th largest commercial bank in the U.S., according to *Fortune* magazine. It has assets of $18.2 billion and operates in 18 other countries.

For many years, Irving Trust has used an account analysis system to keep track of the types of transactions made by the bank's customers. This batch system produces monthly reports listing the volumes of the various kinds of transactions by product line and by customer. Analysts use these reports to study the profitability of the bank's services.

The system was written in COBOL in the mid-1960s. Since then, it has migrated from IBM's disk operating system to its present operating environment, MVS, without changing much. In its monthly run, it manipulates large volumes of data; this run requires six hours on an IBM 3033.

In 1980, data processing was asked if the system could be substantially updated. The system was not serving the bank's needs adequately, and it had become difficult to maintain. For one thing, it collected data from many other systems—demand deposit accounting, letter of credit, electronic funds transfer, and so on. Each time one of these services was changed, or a new service was offered, a new program had to be written for the account analysis system, to extract the relevant data.

ISSN 0012-7523.

The account analysis system had been operating on a monthly batch basis. The users wanted it changed to an on-line information system, through which they could obtain ad hoc information, as well as receive regular reports.

Data processing was willing to update the system if the users could fully and accurately describe their needs. But the users were not sure what they wanted. They knew what they did not like about the old system, but they did not know all the features they would like in the new system.

After some effort trying to create specifications for the new system, the users decided on an alternate approach. They went to the bank's newly formed Information Center and asked if it's people would help them develop the new system.

The Information Center shares an IBM 3033 computer with the software development area. The center seeks to encourage users to do some of their own computing. As an aid, it offers FOCUS™, a data management system from Information Builders, Inc., of New York City (Reference 1), as well as other end-user products.

Since the traditional approach for replacing the account analysis system (by first creating detailed specifications) had not succeeded, the people at the Information Center agreed to help the users replace the system themselves. Several FOCUS consultants were assigned to assist the user team, and the users were given terminals for on-line access to FOCUS.

The users were also told about two utilities—developed by data processing and offered by the Information Center—that would help them. One is a data delivery system that allows users to request data from a production computer. The data is transferred to the development computer within one day's time. The other is a data distribution system that allows users to output large amounts of print data to a disk system. Such reports are then produced once a day.

The project took about 15 months to complete, with the team working part-time on it. During the first four to five months, the team recreated the old system, using FOCUS, and incorporated new features they knew they wanted. When they added a new feature, they allowed other people in the department to test it. This process enabled the team to create a system that was well received, but it also caused development to take longer than expected.

Conversion to the new system was accomplished in three phases. In the first phase, the two systems were run in parallel, with the old system generating a magnetic tape that was used to transfer data to the FOCUS database. This phase lasted about three months.

In the second phase, parallel processing continued, but the tape and card input was loaded directly into the FOCUS system. This phase lasted another three months.

In the third phase, the old system was discontinued.

The users hope to update the input process further, phasing out magnetic tapes and cards. The new system will then draw its inputs directly from the production databases. At this point, however, the manager of the Information Center is not sure whether tapes and cards can be completely eliminated.

Before undertaking development of the new account analysis system, the user team had developed several small systems, using Information Center tools. In effect, they had served as a small application development team within their own department. When they began developing this new system, however, they were surprised how much more difficult the development effort was. Although the coding was easy, they ran into many unexpected procedures—procedures common to most data processing projects. For example, the new system needed data security features, which these users had not anticipated. In addition, the system may eventually lead to direct customer billing for services. So it had to be scrutinized by internal auditors and data processing. The users had no strong arguments against these requirements, they just had not foreseen them.

Also, due to the large number of transactions handled by the system, the users found testing more difficult than they had expected. In order to reduce processing time, they divide the database into many smaller databases—creating twelve monthly transaction databases, a product code database, and a customer account database.

Co-ordinating these databases and writing procedures to update them was also harder than they had anticipated.

Even so, the development methodology the users employed was the only appropriate approach for replacing the old system, because they did not know what they wanted until they had experimented with new features. And they created—and are maintaining—a system that now fills their needs.

Will the Information Center encourage other users to replace old applications in this manner? The answer is: Yes for small systems; probably not for systems with large databases, although the decision may rest with the users. The new monthly batch run is not as efficient as data processing would like, due to the large number of transactions processed. On the other hand, the system is well suited for ad hoc reporting. And it is leading to new and better uses of the data by other departments in the bank.

General Mills, Inc.

General Mills, Inc., with headquarters in Minneapolis, Minnesota, is a diversified corporation with entities in five different industries. They started out in the milling business, and in recent years the company has diversified into other industries, such as toys, specialty retailing, fashion, and restaurants. The company is the 80th largest corporation in the U.S., according to *Fortune* magazine, with annual sales of over $5.3 billion.

A few years ago, the consumer foods group decided that several of their old financial applications needed updating. In addition, two other corporate accounting systems required replacing. These old systems were originally written for their Honeywell computer, in the late 1960s, or their Burroughs computer, in the 1970s. Some of these old systems had been updated, but others had not. Therefore, additional code conversion was required. These systems had become complex and required continual maintenance to stay abreast of the changing business environment and new regulatory agency requirements.

Their approach was to study the marketplace, choose the most appropriate software, and then obtain the necessary hardware—which turned out to be an IBM system. As Snyders briefly describes in Reference 2, the company did not want to expand its data processing staff, even for so large a project; yet they wanted a tailored system. To meet both needs, they acquired both the Corporate General Ledger™ package and the Generation Five™ application generator from American Management Systems (AMS), of Arlington, Virginia.

The software. Several of AMS' products have been written using their Generation Five application generator. Purchasers of these packages automatically obtain the generator to modify, enhance, and maintain the packages.

Generation Five contains: (1) an on-line screen generator, (2) database management facilities, (3) a procedural language that employs accounting terms (such as post and summarize), (4) an accounting-oriented report writer, and (5) precoded routines for common accounting functions. Also, it provides a development methodology, giving all applications the same structure.

For more information about AMS products, see Reference 3.

The application. The basis for General Mills' new integrated system would be the Corporate General Ledger package. Modifications and extensions would be written in Generation Five. The general ledger package had been written using Generation Five, so such changes would fit in with the existing package structure and code. General Mills liked this approach of being able to make modifications within the 'envelope' of the package.

The entire replacement project—replacing all nine systems—took fifteen months, including three months of live parallel processing between the old and new systems. The project could have taken less time, but the company did not want to rush conversion.

Conversion was a joint effort between General Mills and AMS. The General Mills team consisted of one person from data processing, four from consumer foods, and one from corporate accounting. This team concentrated on system design (performed with AMS personnel), data conversion, and system testing. The corporate accounting team member wrote several custom reports using the AMS report writers.

At the same time, AMS had a team of analyst-programmers tailoring the general ledger package, using Generation Five. The General Mills team observed this coding process carefully, so that they could take over maintenance and enhancement when the project was completed.

If the company had opted to redo the old systems using COBOL, the conversion would have become a multi-year effort, costing over a million dollars, the manager of the project told us—because the lengthy time requirement would have been complicated by personnel turnover and the changing business environment. Use of Generation Five reduced the complexity of the effort and speeded up development.

On the question of data conversion, the General Mills team decided not to convert their master and ledger data directly. However, they did want to edit and validate all data coming from the old systems. Therefore, they wrote one-time programs to convert the old systems' data into the package's 'add table maintenance transaction' format. The transactions were then processed through the package's table maintenance routines. Historical ledgers were converted to financial statement transactions and validated against the new validation tables, creating new ledgers.

This one-time validation process served a number of useful purposes. First, it tested the logic in the package's file maintenance routine. Second, it identified discrepancies in the old data and new structure. And third, it uncovered several data problems in the old systems.

To simplify processing, General Mills uses the package's standard file format. All ledgers—whether actual, budget, allocation, or consolidated—use the common format. For special reporting, or interfacing with other applications, they use Generation Five to generate a file with the needed special structure, based on the standard file format.

A recommendation General Mills has for companies considering using application generators is—do a thorough job of planning. On the one hand, application generators may be appropriate for some applications, but not others. Generation Five is written specifically for financial applications. It may not be appropriate for inventory control applications, we were told.

On the other hand, the use of a generator can be abused, unless standard practices are established at the outset. It may provide flexibility that can be misused—it may offer several ways to perform a function, yet there is probably one best way. Together, General Mills and AMS decided on appropriate refinements for the new financial systems and subsequent use of Generation Five.

General Mills' use of the general ledger package and Generation Five has provided several benefits to data processing and the users. For data processing, the large conversion was completed without hiring more staff. In addition, the new system has eliminated the formerly complex 'closing' process, at the end of each month and fiscal year. Data processing now only executes a pre-defined schedule, mounts tapes, and distributes reports. And finally, the new process is efficient. Since application features are integrated, less complex, and more streamlined, they require less processing time.

To the users' benefit, the new system 'belongs' to them. They perform the maintenance. Changes generally are handled via master table maintenance, not re-coding. Enhancements require using Generation Five, which the users have learned. In all, they manage the use of the integrated system, and are responsible for keeping it up-to-date. Product updates are supplied by AMS with an installation guide.

All in all, General Mills is pleased with the way they replaced their important, yet obsolete, financial systems.

The penalty of easy migration

'Easy migration'—it's the catch-word of all computer manufacturers. It means installing new equipment and, at the same time, allowing existing programs to continue running. As we say in the U.S., "It's like apple pie, motherhood, and baseball." How could anyone be against it?

Well, for years, companies have discovered that easy migration has a negative side as well—it chains them to the past. Although the hardware is updated gradually, the software is not; it stays with the old technology. Vendors make it

easy to move onward and upward to more powerful, newer computers by allowing you to drag all of your old programs along with you—not to mention your keypunches, card readers, and other old peripherals these old programs may require. There are companies running programs on their IBM 3033 which were originally written for their IBM 1401 computer—carried forward from the 1960s!

The common retort is, "Why replace these applications? They may be old, but they're running!" The answer is: they are probably costing your company more than you think.

These old programs have caught up with the companies that have long ignored them. Maintenance in many organizations is 70% of the programming effort. And the programs that are hardest to maintain generally are the ones that have been around the longest. They have little documentation, and most of that is practically useless, because it is so out-of-date. They are written in old languages, which many young programmers (who are often assigned maintenance work) do not know. And it is just plain hard to find programmers willing to work on such discouraging 'messes.'

Since these old programs are difficult to maintain, they do not get maintained as they should; therefore, they do not meet users' changing needs. Users often do not request changes, because they know it is a long and tedious process to get such changes made. So users are often unhappy with these old systems, because they have so little control over them.

These old programs are burdensome to users in another way—they use out-of-date procedures. Since they are mainly batch oriented, often using card input, turnaround is slow—usually measured in days, not minutes. Each program generally has its own file, and combining data from several files, to create ad hoc reports interactively, is usually out of the question. So, while the current computer may be up-to-date, the users may still be filling in keypunch forms by hand and waiting a day or more for the results. Thus, these old programs are probably making users less productive than they could be with updated versions.

Information systems management often wants to replace these old applications, but they feel trapped. Rewriting these old programs in, say, COBOL would take even more resources away from 'more important' new work.

But there are now products available that make replacement of old programs practical, relatively easy, and cost effective. It is possible to start 'an early retirement program' for these old systems. Since the software has not been improved as it migrated, a concerted effort will be needed to start this replacement program, in companies with huge numbers of old programs. But the benefits can far out-weigh the difficulties, as we will point out throughout the remainder of this report.

Application development is currently undergoing a major change. The old, tried-and-true method of exhaustive systems analysis coupled with 'hand coding' is giving way to prototyping and partially automated programming. Application and program generators and data management systems, which automate much of the coding, can be used for *most* business applications. They can decrease application development time enormously. Many programmers resist these products, saying that they take away the 'art' and fun of programming. But it is becoming clear that hand-coding of many business applications is no longer cost-effective. Replacing old applications with these new products now appears practical.

In addition, application packages have matured to the point where they can be used to replace some old in-house applications, and perhaps bring about standardization at the same time.

Three replacement approaches

So we see three types of products that can be used to replace old hand-coded applications: Data management systems, application and program generators, and application packages. Following are considerations for using these approaches.

Using data management systems

Data management systems (DMS) contain application development tools based around a database management system. They generally contain 'a fourth generation language'—meaning, a

non-procedural programming language—with which end users can create applications and set up data files. Complex business applications, with links to, say, COBOL subroutines, can also be created using these systems; however, programming expertise is generally needed for these uses.

Most articles about DMS discuss using them to create new applications—but these products are equally useful for replacing old applications. To illustrate this approach, we talked with Mary Rich, an independent consultant in El Segundo, California, about her use of RAMIS II™, a data management system from Mathematica Corp. of Princeton, New Jersey (Reference 4).

Rich has been using RAMIS for five years, and has helped a number of companies replace old business applications with RAMIS and RAMIS II applications. Rich told us about a recent project which is typical of the old applications she has replaced.

The application was an installment loan tracking and billing system—a major application in this organization. It was originally written in COBOL in the early 1970s. When a loan was approved, people in the company's accounting department filled in forms by hand to initiate the payments schedule. These forms were keypunched and the resulting cards were merged by hand into the punched card payment file. The payment cards in turn were used to update a master tape for the daily billing run. A second file of cards contained the loan information. But the cards themselves caused innumerable problems—cards got lost or were duplicated by mistake or were mutilated by the card reader.

This system interfaced with a credit system, which authorized credit for the loans. The credit system had been written in RAMIS II a few years ago. The people in accounting noticed that it was interactive and much easier to use than their old COBOL-based batch system. They asked that their installment loan system be replaced with an interactive RAMIS II system.

The request was approved and three people were put on the project—the maintenance programmer, a new, inexperienced programmer, and Rich. All coding was done in RAMIS II, except for developing the input display screens. These were programmed in COBOL using IBM's Display Management System. At the time, RAMIS II did not have a screen formatter; now it does.

The conversion of the basic, pre-specified system took about four work-months, during a six month time period. This included coding and debugging. However, that was only part of the conversion. As Rich pointed out to us, in all the conversions she has worked on, they have always added enhancements—changes the users had wanted for years.

For example, the specifications for the new system did not describe interfaces to other systems. In the old system, as payments were received, forms were filled out by hand, keypunched, and that data was used to update the general ledger. Also, several important functions had not been automated, such as calculating pre-payments and issuing bills for delinquent payments. Users wanted to fix these problems. And some of the major changes required restructuring the database as well; this happened four times. Since RAMIS II allows iterative development, these new sub-systems were added by first prototyping them, then allowing the users to use them for a short while, and then adjusting them until they were suitable.

Adding missing functions, and relating them to existing functions, adds quite a bit of time to a conversion project, said Rich. So, while coding in RAMIS II, or a similar language, takes very little time, figuring out how to add new elements does take some time. But in the end, the users have a much better system.

The conversion took longer than had been expected for two other reasons: (1) all three programmers worked on the project only part-time, and (2) two of the programmers had no previous experience with RAMIS II, so they were being trained on the job.

The new system was based on a structured analysis of the functions performed by the old system, rather than a duplication of the old code in a new language. This approach was fortunate, Rich told us, because they discovered during the re-write that some of the calculations performed in the old system were wrong—and they had always been wrong! The two systems were run in parallel for three months, and whenever a dis-

crepancy was uncovered, it was the old system that turned out to be incorrect.

Rich said she has seen this happen before; RAMIS II systems tend to have fewer errors in them than the systems they replace because the RAMIS II programs are simpler. It is not unusual, she said, for a 100-line RAMIS II program to replace a 6000-line COBOL program. So the simplicity of DMS programs not only speeds up coding but also leads to more accurate systems.

Data conversion was very simple, because it was performed automatically by RAMIS II. RAMIS II read the tape and cards as transaction files and converted them into a pre-defined RAMIS II database, containing some 65,000 records. Restructuring the database was equally easy.

The organization has received an important benefit from the new system—quick response to needed changes. Due to the recent volatile nature of the financial community, new types of loans have occurred literally overnight. In one rush case, the project team was able to make the needed changes to 40 RAMIS II programs in only four hours. Rich commented that it would have taken longer than that just to re-compile the equivalent COBOL programs!

Users are happier with the system, and it continues to grow. More uses are being found for it, so enhancements are continually being requested. Even with this increased number of requests, there is less total maintenance on the system, because it contains fewer lines of code, has more validation and control features, and is more accurate.

In other conversions, Rich has encountered a few data file conversion problems. These occur only when the old files do not contain valid data or are not well documented. When converting from another database management system, the files usually must be converted from the former DBMS structure to a sequential format, before being read into the RAMIS II database as transactions.

Rich also pointed out a feature of RAMIS II that is a significant aid in any conversion—to define production system files and certain other DBMS files as 'external' files, and then use these with the RAMIS II reporting and data extraction language. For example, it allows programmers to: (1) extract data from production files for testing, (2) produce the same set of reports from the new and old files for verification, and (3) quickly research problems in the data or in the code.

Rich recommended using a DMS for all business applications *except* those that require real-time updating, multiple terminal updating, or very high volume. Applications especially suitable for DMS are those subject to rapid changes (as in a regulatory environment) or where the need for ad hoc reporting is high, as in personnel or budgeting systems.

She also recommended that new RAMIS II programmers spend time doing system analysis work, before they begin coding (although not the exhaustive systems analysis needed when applications are going to be hand-coded). Experienced RAMIS II programmers can perform analysis while prototyping, but inexperienced programmers can get themselves into problems if they forego analysis, thinking that prototyping makes this unnecessary.

Using application generators

Application generators are being hailed as the next step in application development tools, following high level languages, such as COBOL. This may be true, but only if the chosen product matches the application closely. Some of today's application generators are aimed at specific types of applications; they make development of such programs faster and easier, but they will not do the same for all applications. An example is generators for financial applications; they usually would not be appropriate for creating (say) market research systems.

When the generator does not closely fit the application, programmers will revert to their old coding methods, to circumvent the product's limitations. And the company will end up with a hybrid application that was not much faster to develop nor very easy to maintain. So matching the generator to the application is necessary for realizing productivity gains.

As Stewart discusses in Reference 5, there are two types of applications generators. *Program* generators create source code, yielding stand-alone programs. *Application* generators require that a run-time portion of the generator be used

to run the application programs; they do not produce stand-alone code. The distinction may or may not matter. If the system produces source code, statements can be modified directly, without using the generator. However, systems that depend on a run-time module may run faster, says Stewart. Both types generally require programming expertise.

In a presentation at the National Computer Conference last year, Waldrop (Reference 6) discussed the pros and cons of a financial application generator that his company used.

On the benefits side, he said, several of the product's features contributed to the team's ability to finish the large project almost on time.

First, the product had a table-driven architecture. This significantly speeded up development, because all programs could draw on these tables. Changes usually needed to be made only to tables, not to all of the affected programs.

Second, the generator enforced a structured methodology and standards for coding and data names. These, in turn, encouraged the use of common code.

Third, since the generator was designed for financial applications, a number of routines based on common accounting functions contributed significantly to faster development. Such functions as 'reject-batch' and 'reject-document' did not need to be hand-coded, for instance.

Fourth, the generator's automatic transaction suspense file—where transactions are stored until validated—eased the requirements for complex coding for posting transactions to the master file.

Fifth, the generator required only one set of code for processing data in either batch or on-line modes.

And sixth, the screen generator allowed the company to create the on-line system with programmers who had not had prior tele-processing system experience. Since the application required 53 screen formats, the screen generator contributed notably to speeding development.

The company also ran into some unexpected problems—usually caused by unrecognized limitations in the product. To name a few, the product used a non-standard file access method, it did not provide the ability to create interactive display screens on-line (only input-only or output-only screens), and there were constraints on the number of data elements in a control table record.

Since the team had not used the generator previously, they did not design the system with the generator's capabilities and limitations in mind. Thus, when design issues arose, only two (unsatisfactory) solutions were possible. One, the programmers could resort to coding the routine in COBOL; this introduced extra complexity into the system. Or two, the generator was used; this produced a less efficient solution.

Despite these problems, the company was able to write 285 programs in about three months time, with twenty-five people at the peak point. A tribute to the generator, Waldrop stressed.

One point stands out. Today's generators are not as flexible as the languages programmers are accustomed to using. Therefore, to make most efficient use of a generator, system designers must keep the product's capabilities in mind.

Using application packages

We talked with the people at McCormack and Dodge Corporation, of Needham Heights, Massachusetts (Reference 7), about their experiences with companies that have replaced old applications with one of their packages.

They told us they have noticed two differences between users who are replacing a manual system and those replacing an old, automated system. Users replacing a manual system are generally more fearful of any new system—they are afraid they will lose their jobs or they will not be able to cope with the new work environment. Users who are already working with an automated system are generally more accepting of change.

Second, users replacing old, automated systems are usually more willing to compromise on features. They will forego some desired features in return for getting a new system more quickly. They are more realistic in their demands. They generally do expect, however, that a schedule be drawn up for adding missing features. McCormack and Dodge recommends that no changes be made directly to a package; such enhance-

ments should be added through external routines.

Usually it is the users, not information systems people, who urge investigating packages. They are upset that an old application no longer meets their needs, and they do not want to wait in a multi-year backlog queue for in-house development. If the information systems department agrees to consider purchasing an application package, McCormack and Dodge recommends setting up an evaluation team of about four people, representing both users and the department.

The first task of the team is to study the current operation and decide what functions the new system should perform. The team should question whether the existing procedures and output are necessary, or whether they could (or should) be changed. Work flow changes are often necessary when installing a package, the company told us, so the question of whether users are willing to change the way they work is an important issue when considering packages.

With a list of necessary and desirable requirements, the team can begin looking at packages. A package that meets 80% of all requirements—plus *all* of the necessary requirements—should be considered a good fit.

If the company chooses a package that can be customized, the next step is learning how to 'tailor' the package to their needs. All McCormack and Dodge packages use a control file, which contains the package's various options. In order to understand the available options, McCormack and Dodge recommends that the evaluation team attend their one-week training school.

During this week, attendees are taught the details of the package, and they are given the assignment of setting up a 'dummy corporation' and implementing various types of options. Attendees from one organization are kept together as a team. They are encouraged to set up a dummy corporation that operates as their real company operates, so that when they finish the training class, they will have tested many of the options they believe they will want.

Following the week's training, most purchasers feel confident to tailor a package on their own. McCormack and Dodge provides a 'hotline' service, and evaluation team members often call and ask about implementing specific options. And, at times, users request a one-day on-site evaluation by a McCormack and Dodge consultant. But, generally, evaluation teams handle the installation on their own.

Installation typically has four phases. In the first phase, the team creates the control file by selecting the options they want. They then run the package against a 'sample' corporation, supplied with each package. This step ensures that the package operates correctly in their operating environment.

Second, they substitute company files for the sample corporation's files, in a test mode. Generally, files need to be converted to the package's file format. McCormack and Dodge recommends this be done by running the current files through the package as input transactions. This process not only converts the data into the proper format but it also identifies problems—incomplete records, inconsistent data, etc. Package documentation helps users perform this conversion step.

Third, the old and new systems are run in parallel.

And fourth, the company converts totally to the new system. Of course, users need to be involved in the process, to ensure that they change their operating procedures as the conversion moves through these phases.

In the financial area especially, we found numerous 'tailorable' packages. With financial considerations changing so often, due to the more volatile business climate and government regulations, such packages may be a suitable answer for replacing many old, and probably hard-to-maintain, application systems.

Replacement program considerations

Given that the stockpile of old applications has become burdensome, and that a replacement program is needed, what problems will information systems management face when it initiates its 'old application replacement' program?

Programmers will resist the new methods. It appears that programmers and information system managers are not the ones pushing for 'fourth generation software.' Most prefer to stay with

the procedural languages with which they are familiar. It is generally the users who are encouraging information systems departments to install application generators, data management systems, and application packages, we have been told. So, yes, management will face programmer resistance.

Programmers say that using these new programming tools requires less skill and knowledge, so that they (the programmers) will have less value in the market place. And they contend that these automated tools will take the creativity out of their jobs, turning them into mere coders.

Due to the reluctance of programmers to abandon their procedural languages, the job of replacing old applications using these new tools may fall to the enthusiastic, and discontented, users. However, this may not be the wisest choice for replacing *all* systems, especially the large ones. There are numerous non-obvious requirements for many large application systems (such as security, audit, and interfacing with other applications). End users may not know about such considerations, and their systems will most likely be lacking in these aspects. So replacement of old applications probably should not be left entirely up to end users.

Most programmers dislike maintenance work, because it disrupts new, more exciting projects on which they would prefer to work. To help overcome this negative attitude toward maintenance work, management could use a planned replacement program as a new opportunity—to train new and willing programmers on fourth generation languages, while reducing the maintenance burden at the same time.

Experienced programmers are not likely to eagerly accept this new educational opportunity. However, some employees in other departments, as well as new, younger programmers, may be interested in working with users and in learning a fourth-generation language. A conversion project would provide some business training as well, because these programmers would be spending less time on coding and more time on understanding applications.

So a project to replace old applications by using new tools and prototype development techniques can provide a way of training a new generation of programmers, while reducing the application maintenance burden. But it probably will not convert many long-time programmers to using more automated tools.

The new programs will be inefficient. Some people contend that programs created by application generators are inefficient to run. In some cases this is true, says James Martin in his book, *Application Development Without Programmers* (Reference 8), especially where the generator creates source code which must be compiled. But in other cases, the generated code is more efficient than most hand-coded routines, because it is in assembly language or highly optimized code. And it may take less time to compile than hand-written code, if it contains large blocks of pre-compiled code.

However, says Martin, this argument is specious. Only 2% of the business applications in most organizations use 50% of the machine cycles. And 50% of the applications use only 2% of the machine cycles. For this latter 50% of the programs, programmer productivity is far more important than run-time efficiency. Hand coding of most business applications is just not cost-justified any longer, he contends.

Replacement will take longer than planned. When we began our research for this report, we started with a few assumptions. One was that since application generators and data management systems speed coding, they also shorten replacement time considerably. That assumption has proved to be true, but not quite as we anticipated. Many applications can be re-coded in (say) two months time—but the overall elapsed time is often much longer than that.

Why does replacement take longer than expected? For one thing, when using prototyping, much experimentation takes place, as users try out new features. To give users time to test out the basic replacement system (and new features as they are added), replacement projects are often scheduled as part-time activities for several people. Therefore, applications are generally not replaced in a short, full-time effort, but rather through a more drawn-out part-time process.

Another reason it takes longer than expected is that there is almost always little or no useful documentation of the old system. Even the code is often difficult to understand in old applications, especially if a 'bowl of spaghetti' control structure has evolved. So replacement is generally not a case of copying old code line-for-line. Quite a bit of analysis is needed to first understand the old system, find out what changes users want, and then prototype a new system that incorporates the new, desired features. Also, time is needed for analysis, for testing the code, and for testing the interfaces to other applications. Creating the new system can take longer than expected.

Also, we gather that companies do not have guidelines for prototype development using application generators, since the concept is so new. Thus, there could be wasted efforts. For example, too little time may be spent designing the database structure, so it must be redone several times. Refining parts of the system is inherent in the prototyping process, but some rework may become unnecessary with more forethought.

Therefore, a program of planned replacement should have some guidelines that make prototype development more efficient, and some tools to help the developers unravel old systems as well.

While replacement with these new tools is faster than with conventional methods, it does not happen 'overnight.' However, it is important to note that ease of coding and *ease of making changes*, to data structures and program code, allow companies to now consider replacing applications which they would hesitate to replace with conventional programming methods.

And there is another factor to consider: The longer that replacement of old application systems is delayed, the worse the maintenance problem will get for such systems.

The new tools are inappropriate for our applications. Many argue that application generators and data management systems are only appropriate for smaller applications; they cannot be used to replace the complex code in their older applications. Martin, and others we have talked with, say this is just not true. Most functions in business applications that can be programmed in COBOL can also be programmed in these newer languages.

To illustrate this point, Martin gives an example of a medium-size bank which had about 452 COBOL programs a few years ago. After studying these programs, Martin stated that if those applications were being developed today, rather than in the early 1970s, the bank would need only 25 COBOL programs. Most of the remaining programs could be coded using a combination of: (1) a database management system that contains an application generator, a report generator, and a query language, and (2) purchased packages. The bank's systems which were appropriate for these approaches included account checking, domestic savings, trust accounting, domestic loans, domestic mortgage, general ledger, and others, says Martin.

Several other applications could be moved off the mainframe onto a mini-computer with a DMS, says Martin, further reducing their dependence on hand-coding. In this category, he includes their systems for credit and collection, safe deposits registration, cable and mail control, and stationery and supplies control.

Looking at business applications from a slightly different view, Martin places them into five categories. (1) Small application programs that are less than 2000 lines of basic assembly language code. About 70% of most organizations' business application programs fall into this category, he says. (2) Small-to-medium size application programs that range from 2000 to 16,000 lines of code. About 20% of most organizations' business applications fit here, he states. (3) And medium programs that range from 16,000 to 64,000 lines. Only 6% of most organizations' programs are this large. So the great majority of business applications fit within the lower two categories (less than 16,000 lines of code), says Martin, and the 64,000 limit covers 96% of all programs.

The top two categories—(4) large and (5) super-large—contain programs written mainly by software houses and computer manufacturers. They account for only 4% of the business applications of most organizations. Martin believes they should be purchased packages rather than

written in-house, because they are so large and complex.

Using these classifications, Martin says that application generators can be used in small and small-to-medium size applications—which account for about 90% of most organizations' business applications. For those applications which contain complex logic, procedural language routines can be hand-written and combined with the generated code. In addition, to take advantage of database management systems for these applications, prior database designing and planning is needed.

However, it is fair to say that not all old applications should be replaced with these new tools. On-line production systems, in particular, which handle large volumes of transactions, may not be appropriate if performance is a crucial factor in their usefulness.

As mentioned earlier, applications which should be considered for replacement using these new tools are those with frequently changing needs, where ad hoc reporting is important, and where users want to replace a batch system with an on-line information system.

In conclusion, we suggest that companies look at the three types of tools discussed, with an eye toward replacing old applications. Maintenance costs are often way out-of-hand. Companies need to start chipping away at the sources of this problem. One way is to begin replacing some of the more obsolete and burdensome programs, and train new programmers in the new development methods at the same time.

REFERENCES

1. For more information on FOCUS, contact Information Builders, Inc., 250 Broadway, New York, New York 10001; tel. (212) 736-4433.
2. Snyders, Jan, "Generators overcome programmer shortages," *Computer Decisions*, Hayden Publishing Co. (50 Essex St., Rochelle Park, New Jersey 07662), March 1981, pp. 34-42; price $4.
3. For more information about American Management Systems and their products, contact them at 1777 North Kent St., Arlington, Virginia 22209; tel. (703) 841-6000.
4. For more information about RAMIS II, contact Mathematica Products Group at P.O. Box 2392, Princeton, New Jersey 08540; tel. (609) 799-2600.
5. Stewart, George, "Program generators," *Popular Computing*, (70 Main Street, Peterborough, Massachusetts 03458) September 1982, pp. 112-122; price $2.50.
6. Waldrop, James, "Application generators: A case study," *Proceedings of the 1982 National Computer Conference*, AFIPS Press (1815 North Lynn St., Arlington, Virginia 22209), 1982, pp 351-358; price $75.
7. For more information about McCormack and Dodge and their products, contact them at 560 Hillside Ave., Needham Heights, Massachusetts 02194; tel. (617) 449-4012.
8. Martin, James, *Application Development Without Programmers*, Prentice-Hall, Inc. (Englewood Cliffs, New Jersey 07632), 1982; price $32.50.

Prepared by:
Barbara C. McNurlin
Associate Editor

Most of us take for granted the way we and our organizations work. However, the use of new information technologies is gradually changing that. Next month we describe how the work environment is changing at four companies, through their use of several types of new information systems.

COMMENTARY

This month, our Commentary is a selective listing of data management systems and application and program generators for mainframes, minis, and micro-computers. A number of these products can be used by employees who have no programming experience, to do their own programming. Where no specific type of mainframe or mini-computer is specified, generally such products are written in ANS COBOL. We have been compiling and updating this listing for several years; while we believe the list is accurate, some changes may have occurred of which we are not aware.

ACEP, Software Module Marketing, 1007 7th St., Sacramento, California 95814. For IBM computers. Tel: 916-441-7234.

ADF, DMS, QUERY BY EXAMPLE, GIS, and System 38 facilities, contact IBM at your local sales office. For their various computers.

ADF/ASDM, M. Bryce and Associates, Inc., 1248 Springfield Pike, Cincinnati, Ohio 45215. For mainframe and some mini computers. Tel: 513-761-8400.

ADS and ON-LINE ENGLISH, Cullinet Software, Inc. (formerly Cullinane Database Systems, Inc.) 400 Blue Hill Dr., Westwood, Massachusetts 02090. For mainframe computers. Tel: 617-329-7700.

AIMS 3, Aims Plus, P. O. Box 17247, Austin, Texas 78760. For Wang computers. Tel: 512-385-0702.

ALL and ENGLISH, Microdata, 17481 Red Hill Ave., Irvine, California 92705. For their systems. Tel: 714-540-6730.

ANSWER/DB, INQUIRY, IV/IMS, and MARK V, Informatics General Corp., 21050 Vanowen St., Canoga Park, California 91304. For mainframe computers. Tel: 213-716-1616.

ASI-ST, Applications Software, Inc., 21515 Hawthorne Blvd., Torrance, California 90503. For mainframe computers. Tel: 213-540-0111.

AUTOPILOT, Software Automation Inc., 1100 Business Parkway, Richardson, Texas 75081. For small machines. Tel: 214-231-9142.

BASIS, Battelle Columbus Labs., 505 King Ave., Columbus, Ohio 43201. For CDC, Univac, IBM, and DEC systems. Tel: 614-424-5524.

BOP, Production Engineering Laboratory, Nth Sintef, N-7034 Trondheim-Nth, Norway. Prototyping tool for factory management systems. Tel: 075-93-800.

CENTRAL SOFTWARE, Planning Research Corporation, 1500 Planning Research Drive, McLean, Virginia 22102. Tel: 703-556-2200.

COBGEN, Software Info Services, Inc., 3600 Wilshire Blvd., Suite 1510, Los Angeles, California 90010. COBOL program generator for IBM OS and DOS systems. Tel: 213-380-0466.

COGEN, from: (1) Bytek, 1714 Solano Ave., Berkeley, California 94707, for any Ryan-McFarland COBOL system; and (2) Key Microsystems Inc., 822 Boylston St., Chestnut Hill, Massachusetts 02167, for micro-computers using CP/M 80 and 86, and MP/M 80 and 86.

CPG, Insac Software Inc., 2300 Peachford Road, Atlanta, Georgia 30338. For IBM mainframe computers. Tel: 404-452-7676.

CREATE, Complete Computer Systems, 159 Gibralter Road, Horsham, Pennsylvania 19044. For Data General systems. Tel: 215-441-4200.

CUPID, Time Utilising Business Systems, 30 New Walk, Leicester, United Kingdom. For PDP-11s.

DATASCAN, Data Management Systems Inc., 4360 Georgetown Square, No. 809, Atlanta, Georgia 30338. For IBM System 34 and Datapoint computers.

DATATRIEVE, Digital Equipment Corporation, 100 Main St., Maynard, Massachusetts. For their equipment. Tel: 617-897-5111.

dBASE II, Ashton-Tate, 9929 W. Jefferson Blvd., Culver City, California 90230. For 8-bit CP/M systems. Tel: 213-204-5570.

DRB Program Generator, David R. Black and Associates, 1780 Maple Ave., Northfield, Illinois 60093. For a number of mainframe and mini-computer systems. Tel: 312-441-8122.

EASYTRIEVE and OWL, Pansophic Systems Inc., 709 Enterprise Drive, Oak Brook, Illinois 60521. For mainframe systems. Tel: 800-323-7335.

FIRST, Data General Corp., 400 Computer Dr., Westboro, Massachusetts 01580. For their computers. Tel: 617-366-8911.

FOCUS, Information Builders, 1250 Broadway, New York, New York 10001. For mainframes and on time-sharing services. Tel: 212-736-4433.

FOUNDATION, Applied Data Research, Route 206 at Orchard Road, Princeton, New Jersey 08540. For mainframes. Tel: 201-874-9100.

GENASYS, Genasys International Inc., 17 East 45th St., New York, New York 10017. For mainframe computers. Tel: 212-687-2015.

GENERATION FIVE, American Management Systems, Inc., 1515 Wilson Blvd., Arlington, Virginia 22207. For IBM and DEC systems; to be used in combination with their application packages. Tel: 703-841-6000.

HERO, The Software Authority, 1270 Oakmead Parkway, Sunnyvale, California 94086. For IBM-compatible mainframes. Tel: 408-749-9100n

IFM and INFORM, United Computing Systems Inc., P. O. Box 8551, Kansas City, Missouri 64114. Available on their time-sharing service.

INFO, Henco, Inc., 35 Walnut St., Wellesley, Massachusetts 02181. For IBM, Honeywell, and Prime systems. Tel: 617-237-4156.

INFORMATION, Prime Computers, Prime Park, Natick, Massachusetts 01760. For Prime equipment. Tel: 617-658-8000.

INGRES and MicroINGRES. Relational Technology, Inc., 2855 Telegraph Road, Suite 312, Berkeley, California 94705. INGRES runs on DEC VAX computers, and MicroINGRES is available for some micros using the Motorola MC68000 processor. Tel: 415-845-1700.

INQUIRE and IQ/NET, Infodata, 5205 Leesburg Pike, Falls Church, Virginia 22041. For IBM equipment. Tel: 800-336-4939.

INSYTE, Response Technology, Inc., 4064 South West Donovan, Seattle, Washington 98101). For large Burroughs machines. Tel: 206-937-7845.

LINK, Burroughs Corporation, One Burroughs Place, Detroit, Michigan 48232. For their machines. Tel: 313-972-7350.

MANAGE, Computer Sciences Corporation, 650 N. Sepulveda Blvd., El Segundo, California 90245. Available through their Infonet time-sharing service. Tel: 213-615-0311.

MANTIS, Cincom Systems Inc., 2300 Montana Ave., Cincinnati, Ohio 45211. For mainframe systems. Tel: 800-543-3010.

MAPPER and ESCORT, Sperry Corp. Box 500, Blue Bell, Pennsylvania 19424. For Univac machines. Tel: 215-542-4211.

MARS, J and S Associates, 1345 Wiley, Schaumburg, Illinois 60172. For Honeywell DPS 6 mini-computers. Tel: 312-882-2878.

MDBS III, Micro Data Base Systems, Inc., Box 248, Lafayette, Indiana 47902. For 8- and 16-bit micros, CP/M and other operating systems, plus PDP-11 under UNIX or RSX-11M. Tel: 317-742-7388.

MICRO AP, Micro AP, Inc., 7033 Village Parkway, Dublin, California 94566. For micro-computers using the CP/M operating system. Tel: 415-828-6697.

MIMS, GEISCO, One New England Executive Park, Burlington, Massachusetts 01803. Available through their time-sharing service. Tel: 617-273-1411.

NATURAL, Software AG of N.A., 11800 Sunrise Valley Drive, Reston, Virginia 22091. For IBM mainframe systems. Tel: 703-860-5050.

NOCODE, General Automation Inc., 1055 South Street, Anaheim, California 92803. For their computers. Tel: 714-778-4800.

NOMAD2, National CSS, 187 Danbury Road, Wilton, Connecticut 06897. Available through their network. Tel: 203-762-2511.

ORACLE, Relational Software Inc., 3000 Sand Hill Road, Menlo Park, California 94025. For IBM mainframes, DEC VAX, PDP-11, and LSI-11 (micro) computers. Tel: 415-854-7350.

PERSONAL PEARL, Computer Pathways Unlimited, 2151 Davcor St. SE, Salem, Oregon 97302. For micro-computers using CP/M. TEL: 505-363-8929.

PROGENI TOOLS, Progeni Systems Inc., 715 North Central Ave., Glendale, California 91203. For Burroughs systems. Tel: 213-956-0251.

QUESTOR, Comshare Inc., 3001 S. State St., Ann Arbor, Michigan 48106. Tel: 313-994-4800.

QUIC-N-EASI, Standard MicroSystems, 136 Granite Hill Court, Langhorne, Pennsylvania 19047. For micros using CP/M. Tel: 215-968-0689.

RAMIS II, Mathematica Products Group, P. O. Box 2392, Princeton, New Jersey 08540. For IBM machines and on several time-sharing services. Tel: 609-799-2600.

READY CODE, Raytheon Computer Services, Wayside Road, Burlington, Massachusetts 01801. Available on their time-sharing service. Tel: 617-272-9300.

REVIEW, Covill Associates, 9528 Miramar Road, Suite 124, San Diego, California 92126. For Burroughs B7000/B6000 computers. Tel: 619-270-6340.

SCORE, Software Design Associates, 260 Madison Ave., New York, New York 10016. For Univac and IBM equipment. Tel: 212-686-2032.

SEQUITUR, Pacific Software Manufacturing Co., 2608 Eighth Street, Berkeley, California 94710. For 16-bit micros. Tel: 415-540-5000.

SMART, Cincinnati Data Systems, 4250 Creek Rd., Cincinnati, Ohio 45241. For mainframe systems. Tel: 513-891-6647.

SPEED II, TOM: The Office Manager, P. O. Box 66596, Seattle, Washington 98166. For Wang VS and 2200 computers. Tel: 206-246-7022.

SUPER ENGLISH IX, Automated Quill Inc., 3501 S. Corona St., Top Floor, Englewood, Colorado 80110. For Data General computers.

SYSTEM-80, Phoenix Systems Inc., 1106 Ohio River Blvd., Sewickley, Pennsylvania 15143. For mainframes, minis, micros. Tel: 412-741-8330.

TAPS/80, Intel Systems Corporation, 12675 Research Blvd., Austin, Texas 78766. For mainframe systems. Tel: 512-258-5171.

THE FORMULA, Dynamic Microprocessor Associates, 545 5th Ave., New York, New York 10001. For micro-computers. Tel: 212-687-7115.

THE LAST ONE, DJ 'AI' Systems Inc., (1) Station Road, Ilminster, Somerset TA19 9BQ, England; tel: 04605-4117; (2) 2 Century Plaza, Suite 480, 2049 Century Park East, Los Angeles, California 90067; tel: 213-203-0851. For CP/M micro-computers.

THE TOOL, High Technology Software Products, 2201 N.E. 63rd, Oklahoma City, Oklahoma 73113. For Apple II. Tel: 405-478-2105.

UFO, Oxford Software Corp., 174 Boulevard St., Hasbrouck Heights, New Jersey 07604. For IBM computers. Tel: 800-631-1615.

USER-BASE, UserWare International, 2235 Meyers Ave., Escondido, California 92025. A micro-computer version of USER-11 that runs under OASIS operating system on both 8-bit (Z80) and 16-bit (MC68000) micros. Tel: 619-741-8825.

USER-11, North County Computer Services, 2235 Meyers Ave., Escondido, California 92025. For larger DEC PDP-11 systems running under RSTS operating system. Tel: 619-745-6006.

VISION, Four-Phase Systems, 10700 North DeAnza Blvd., Cupertino, California 95014. For their systems. Tel: 408-255-0900.

WANG PROGRAM GENERATOR, Wang Laboratories, 1 Industrial Ave., Lowell, Massachusetts 01851. For Wang systems. Tel: 617-459-5000.

SUBJECTS COVERED BY EDP ANALYZER IN PRIOR YEARS

1980 (Volume 18)

Number	*Coverage*
1. Managing the Computer Workload	I
2. How Companies Are Preparing for Change	K
3. Introducing Advanced Technology	K
4. Risk Assessment for Distributed Systems	L,E,A
5. An Update on Corporate EFT	M
6. In Your Future: Local Computer Networks	F,B
7. Quantitative Methods for Capacity Planning	I
8. Finding Qualified EDP Personnel	J
9. Various Paths to Electronic Mail	D,M
10. Tools for Building Distributed Systems	E,B,F
11. Educating Executives on New Technology	K
12. Get Ready for Managerial Work-Stations	C,A,B

1981 (Volume 19)

Number	*Coverage*
1. The Coming Impact of New Technology	K,A,B
2. Energy Management Systems	M
3. DBMS for Mini-Computers	G,B
4. The Challenge of "Increased Productivity"	J,K,A
5. "Programming" by End Users	C,H,B,G
6. Supporting End User Programming	C,H,B,K
7. A New View of Data Dictionaries	G,B
8. Easing the Software Maintenance Burden	H,B,G
9. Developing Systems by Prototyping	H,B,G
10. Application System Design Aids	H
11. A New Approach to Local Networks	F,K
12. Portable Software for Small Machines	B,H

1982 (Volume 20)

Number	*Coverage*
1. Practical Office Automation	A,B,C,K
2. Computer Graphics for Business	K,C,B
3. Interesting Decision Support Systems	C,B,H,A
4. Can Tele-communications Replace Travel?	A,F,J,L
5. The Human Side of Office Automation	A,J,K
6. Some Users Want Their Own Computers	B,C,K
7. Using Minis and Micros	B,E
8. Training for End Users	C,B,K,J
9. Query Systems for End Users	C,B
10. Relational Database Systems Are Here!	G,C,H
11. Experiences with Tele-Commuting	C,K,D
12. Alternate Approaches to Offices Systems	A,B,K

1983 (Volume 21)

Number	*Coverage*
1. Planning Your Future Information Systems	K,B,E
2. Plan Now for Work Stations	K,B,E
3. Replacing Old Applications	G,H

Coverage code:

A Office automation
B Using minis & micros
C Managerial uses of computers
D Computer message systems
E Distributed systems
F Data communications
G Data management and database
H Analysis, design, programming
I Computer operations
J Personnel
K Introducing new technology
L Security, privacy, integrity
M New application areas

(List of subjects prior to 1980 sent upon request)

Prices: For a one-year subscription, the U.S. price is $66. For Canada and Mexico, the price is $66 *in US. dollars,* for surface delivery, and $73 for air mail delivery. For all other countries, the price is $78, including AIR MAIL delivery.

Back issue prices: $7 per copy for the U.S., Canada, and Mexico; $8 per copy for all other countries, sent via AIR MAIL.

Reduced prices are in effect for multiple copy subscriptions, multiple year subscriptions, and for larger quantities of a back issue. Write for details. Agency orders are limited to single copy subscription for one-, two-, and three-years only.

Please include payment with order. For U.S. subscribers, you can use your Visa or MasterCard charge card; include your card name, number, and card expiration date on your order.

For payments from outside the U.S., in order to obtain the above prices, take your choice of three options: (1) use an international money order, (2) pay in U.S. dollars with a check drawn on a bank in the U.S., or (3) use any of the following charge cards: Visa, MasterCard, Eurocard, Access Card, Standard Bank/Kaart, Union Card International, or Diamond Card International. Please be sure to include your card name, number, and card expiration date on your order.

Editorial: Richard G. Canning, Editor and Publisher; Barbara McNurlin, Associate Editor. While the contents of this report are based on the best information available to us, we cannot guarantee them.

Missing Issues: Please report the non-receipt of an issue within one month of normal receiving date; missing issues requested after this time will be supplied at the regular back-issue price.

Address: Canning Publications, Inc. 925 Anza Avenue, Vista, California 92083. Phone (619) 724-3233, 724-5900.

Microfilm: EDP Analyzer is available in microform, from University Microfilms International, Dept. P.R. (1) 300 North Zeeb Road, Ann Arbor, Mich. 48106, or (2) 30-32 Mortimer Street, London WIN 7RA, U.K.

Declaration of Principles: This publication is designed to provide accurate and authorative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional service. If legal advice or other expert assistance is required, the services of a competent professional person should be sought. — *From a Declaration of Principles jointly adopted by a Committee of the Amercian Bar Association and a Committee of Publishers.*

EXECUTIVE SUMMARY

Information systems departments have become over-burdened with maintenance work. One of the causes of this discouraging situation is 'easy migration.' Hardware has gradually been upgraded, but application software has not. Companies have dragged their old programs along with them—programs that are causing more than their share of misery, both to information systems people and to end users.

There are now application development tools available that can aid information systems departments in replacing some of these old programs. Data management systems, application and program generators, and application packages make replacement of some old applications feasible. Applications most appropriate for replacement are those which have constantly changing requirements, those where ad hoc reporting is important, and those where users want an on-line information system to replace their old batch processing system.

Data management systems generally have 'a fourth generation language,' a report writer, a screen formatter, and other utilities, all based around a database management system. These systems are intended for end users (for writing simpler systems) as well as for programmers (for developing more complex systems). They speed coding, ease file creation, and encourage prototyping.

Application generators generate programs by using user-supplied parameters to direct processing of pre-coded routines. Two considerations are important when evaluating generators. One, the generator and the new system need to be closely matched (as we discuss). And, two, the new system must be designed with the generator's capabilities and limitations in mind (for reasons we discuss).

Application packages are becoming increasingly 'tailorable.' Users often can select desired options and store them in a control table. In another approach, one company's packages are written using an application generator so that modifications can be made to enhance the packages, using the generator.

Application development is changing; more automated tools are becoming available. Thus, we think it is time for companies to start an 'old applications replacement program.' Information systems management will face resistance to the new methods. But this new program could train new programmers in the new methods and reduce the maintenance burden as well.