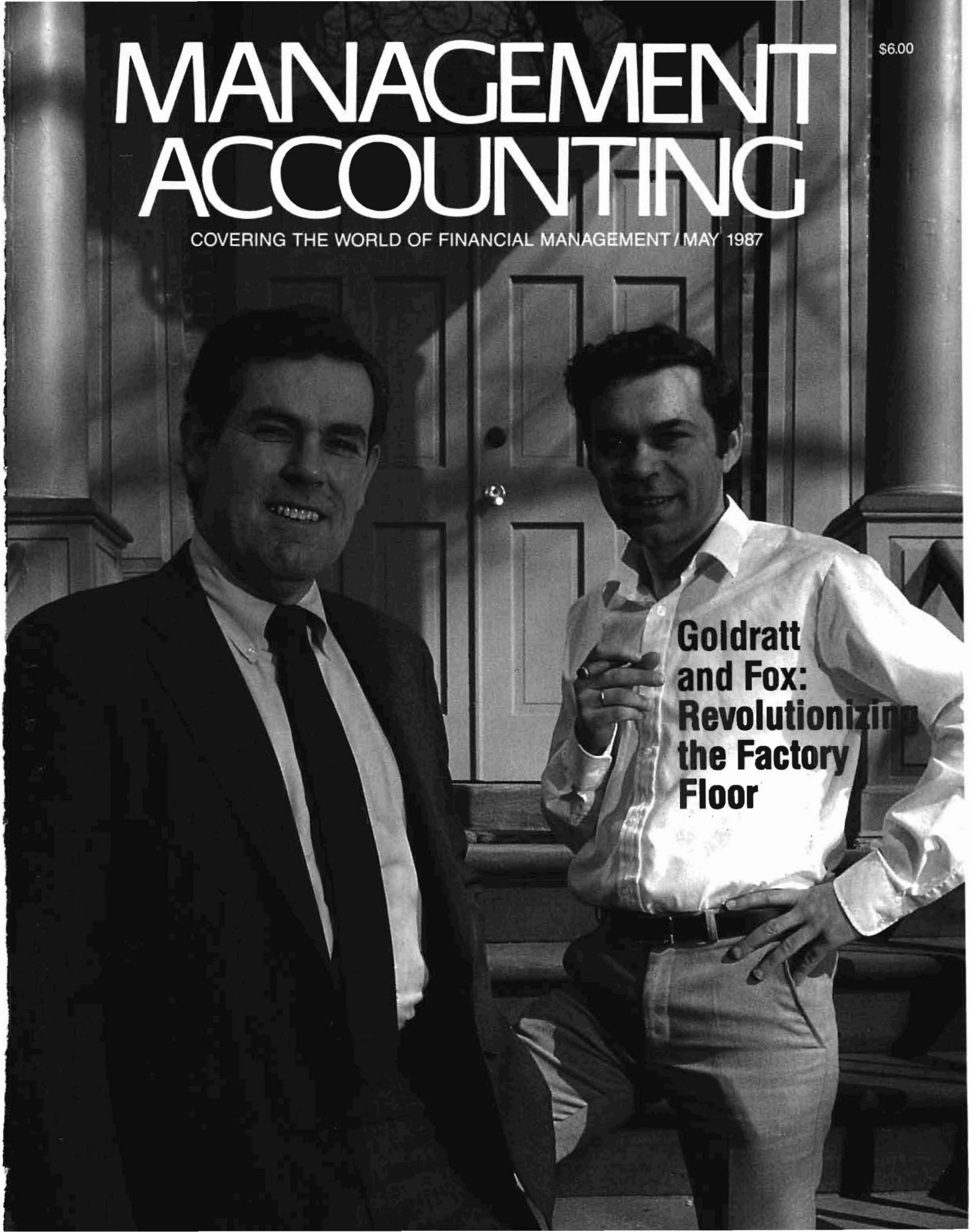
MANAGEMENT ACCOUNTING

$6.00

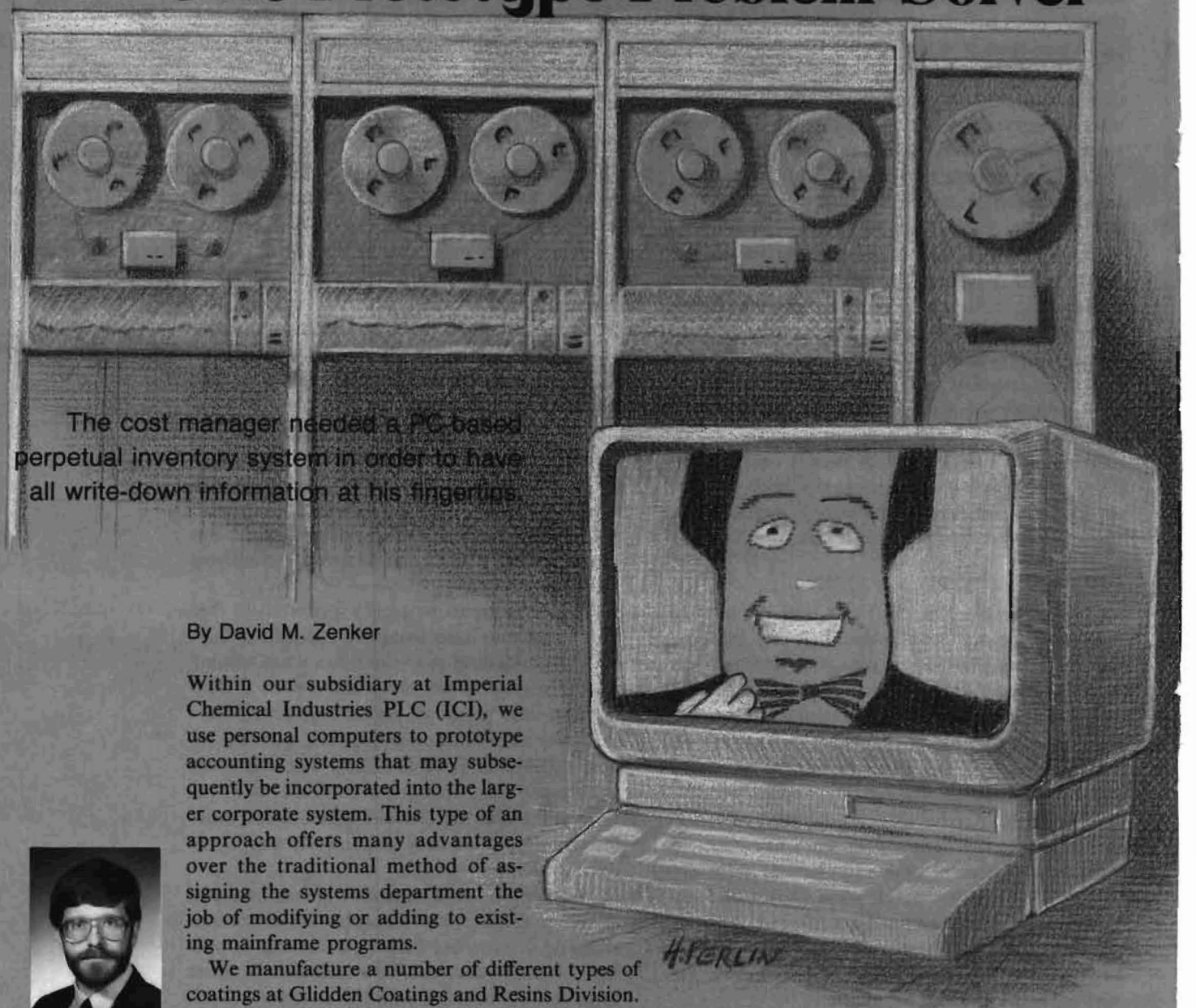
COVERING THE WORLD OF FINANCIAL MANAGEMENT / MAY 1987



Goldratt and Fox: Revolutionizing the Factory Floor

MANAGEMENT ACCOUNTING

The PC as Prototype Problem-Solver



The cost manager needed a PC-based perpetual inventory system in order to have all write-down information at his fingertips.

By David M. Zenker

Within our subsidiary at Imperial Chemical Industries PLC (ICI), we use personal computers to prototype accounting systems that may subsequently be incorporated into the larger corporate system. This type of an approach offers many advantages over the traditional method of assigning the systems department the job of modifying or adding to existing mainframe programs.

David M. Zenker is a senior analyst at Glidden Paints. He is a member of Cleveland Chapter, through which he submitted this article.

We manufacture a number of different types of coatings at Glidden Coatings and Resins Division. One class of our inventory is defined as 186 material, which is material with a six-month shelf life. When this material becomes over six months old, it is defined as distressed and must be written down to ERV (estimated recoverable value).

The problem lies in effectively tracking the product while it is considered ERV material. It must be followed from the time it is written down until it is ultimately disposed of, either through sale, rework, or incineration.

Currently all of our regions are using a manual system to record what material is overaged and what material has been recovered. A standard form, which is generally acknowledged as an interim measure until a better solution can be found, is used to detail most of the necessary data.

A number of bits of information must be gathered, and there is the need to assign responsibility for the material going overage. Responsibility is attributed to either sales, manufacturing, or technical. Depending on the circumstances surrounding the product in question, any one of these functional areas could be held culpable.

Historical information also must be gathered relating to the circumstances of how the material was written down to ERV. For instance, material that is defective but was inadvertently delivered to the customer's location is subsequently returned on an RGT (returned goods ticket). The circumstances surrounding the return must be recorded along with the return of materials. All three areas of responsibility are interested in knowing what went wrong. No department likes to see product returned to the plant nor likes to be held responsible for its return.

Currently, this information is gathered in the

manual system by different people at different times. There is no good cross reference to determine where the historical information lies. As the actual incident recedes into the past, there is no efficient way to resurrect the product history and the associated information that led to its write-down and its final disposition. This makes it difficult for concerned parties to derive common historical information about a product problem and reconstruct how the problem was ultimately resolved.

Each of the three functional areas kept information relating to its area of responsibility, but there was no common repository that would coordinate the whole. The logical area to gather this information is the cost accounting department—the department that has the ultimate responsibility for the accurate valuation of inventory.

Because cost bears this burden, it is paramount that all information relating to inventory return, write-down, and damage due to manufacturing or technical error be immediately incorporated into the inventory accounts to assure accurate reporting of division assets. Because a manager's cost center is going to be affected by the revaluation of inventory, it is also critical that the cost department serve as more than a mere recording mechanism.

Often there will be honest disagreement between managers as to why the material had to be written down, and whose cost center should bear the ultimate responsibility. For this reason, the cost manager should have a position within the organization that parallels those managers that he is expected to mediate.

To mediate effectively, the cost manager must have access to all the information that the other managers have, in addition to a thorough understanding of division policy dealing with inventory revaluations. A perpetual inventory system on a PC could go a long way toward giving the cost manager the information he needs to see that the job is done effectively and equitably.

Selecting the Tools

The system that was chosen to implement this application was the SMART System produced by Innovative Software of Overland Park, Kansas. This package is composed of a data manager, a spreadsheet, a word processor, a time manager, and a communications package.

The communications package holds a special interest for our organization because much of the headquarters reporting that is required in connection with revalued inventory is done manually. If an effective application could be developed, it would be even better received if there were a mechanism to consolidate the information automatically. We hope that SMART's communication capability will provide this mechanism.

The primary component of the SMART System used for the application was the data manager, which uses a virtual memory scheme that is most appropriate for the prototype under discussion. This scheme permits the user to develop models that are only limited by the external memory of the computer. This was an important consideration because some of our regions have a great deal of distressed inventory on hand and because the ultimate consolidation of all of this information will result in a very large file. A data manager that limited its file size to the confines of RAM (random access memory) would be ill equipped to handle the application.

The SMART system also will directly convert LOTUS and Dbase files into the SMART format. This "cross vendor compatibility" is a tremendous benefit to locations that have committed significant amounts of time to inputting information into software other than SMART.

Creating the Model

The person creating the model should be someone very familiar with the problem. It should be someone who is at home with a personal computer and the associated software and has a vested interest in seeing that the project is undertaken and carried to a successful conclusion. He or she should have the time to complete the model and the capability to follow through with the project. It is not probable that the individual that undertakes this, or other significant PC applications, will have an open-ended period of time for completion.

The PC prototype must come down from the ivory tower into the grimy work-a-day world.

There should be sufficient flexibility in the planning stages to allow for the inevitable problems that will crop up. For instance, it might be discovered that additional hardware is necessary to run the application. Perhaps it was originally thought that the application would run on a floppy system and suddenly it has to have a hard drive. Undoubtedly there will be at least one or two false starts.

Unless a very similar application has been previously developed, there will probably be some unique problems that must be overcome. These inevitable problems will stretch the implementation date. The "just one more enhancement" syndrome can cause a project to stretch indefinitely.

The individual who is responsible for developing the model should make every effort to ascertain what all end users want the model to do. When this information is obtained, every effort should be made to incorporate these capabilities into the model at the beginning, rather than retrofitting them when the model is nearly done. If this approach to the project isn't adopted, the project will rapidly progress to the 90% completion point

and then stay there until management abandons it.

Out of the Womb

Once the PC Prototype is complete, it is ready to be tested in real life with real data. It's ready to come out of the womb and into the real world with all the people who don't know or care how much time was spent in its development.

This is the environment that the PC Prototype descends into. From the rarefied air of the ivory tower to the dirty, grimy, "I haven't got time for this" work-a-day world. This is why it is so important to choose your test site carefully.

If a real problem exists, the PC version can keep the problem from becoming overwhelming until systems can address it.

Under ideal conditions, the chief executive officer and the board of directors would be keenly interested in the outcome of the test, and the applications developer would be on hand to address any questions or problems directly. But in the real world, the prototype is going to be sent to a location staffed with people who have other things to do and not enough time to do them in. So, the question remains, what do you look for in a site to test the PC Prototype?

There must be a need. The system must address a problem that is relevant to the people who will be conducting the test. It's very hard to get someone to participate in something that he or she considers nothing but an academic exercise.

There must be a "key participant." Someone in the receiving location must either supervise the operation or take responsibility for seeing that it is implemented properly. This person must have some stake in the outcome. In the best of all worlds, this key participant would be the applications programmer/designer.

There must be support from local management. The key participant must be permitted time to get the operation up and running. Because there will be inevitable mix-ups and probable errors in the initial stages, the person who is delegated to implement the operation must be provided the necessary support to assure that it is a success.

There must be adequate time provided. In the installation of any system of any significance, there will be problems. It may take several months of concurrent reporting to assure that the new system is operating properly and is producing the correct results. If the system is jettisoned before it is given an adequate testing, a great amount of time and effort will have been expended for nothing.

There must be adequate equipment, both hardware and software. Well managed companies desire to accomplish things as inexpensively as possible. This is true of my own. The question becomes, what is the minimal amount that can be expended and still adequately support the project? It is up to the application's developer to articulate his needs effectively to assure that the requisite equipment and software are provided. A savings that paralyzes the project is no savings.

Real World Use

Once the site has been selected and the project has been successfully implemented, inevitable questions will arise in the day-to-day operation of the project. These questions must be answered to the satisfaction of the user so that he will have confidence in the system and not try to bypass it.

It's during this phase that the application's developer finds out if his application is really as easy as it seemed during testing. Once again, in the best of all worlds, the applications developer is also the key participant. If this is the case, all questions can be handled at the implementation site.

If the developer is at another location, he must be prepared to answer questions about the application. The developer must be ready to assuage fears that something is going awry when an unusual happening occurs. Nothing turns a supportive key participant into a hostile user faster than having his data trashed because of a program glitch. Developers will experience three phases:

Extensive hand holding phase. Unless you have a user that is familiar with the problem, familiar with micros, and you have included help screens that are on the order of LOTUS 1-2-3's thoroughness, you can anticipate a period of extensive hand holding. This period is inversely related to how closely your key participant and your application meets the first three criteria detailed above.

Periodic strange occurrence phase. During this phase the user is almost self-sufficient. He can conduct day-to-day operations using the application without assistance. He is capable of doing everything that needs to be done except for those things that are not daily occurrences. For instance, a year-end closing procedure.

There is more than enough support for main corporate systems during year-end, but not for micro systems. For micro systems, the applications developer is still the only game in town. The problems encountered during this period will vary based on the amount of preparation.

If there is an adequate procedure in the documentation, the developer might just point this out. Depending on the key participant's level of sophistication, this might be enough support. It is advisable to call the key participant as the time of the anticipated occurrence approaches. This tactic will engender confidence and emphasize that there is continuing interest in the application.

Manual support phase. This is the final and the most pleasant level of support. In this phase the key participant and others at the site know how the application works. Moreover, participants themselves generally can resolve the infrequent

questions that do arise by using the documentation.

If this system isn't adapted to the main corporate system, it will have to survive without centralized support. It is unlikely that the applications developer will be in the same position forever. This is why it is so important to share the knowledge of operating the system as quickly as time and user capability permit.

The one thing that the applications developer doesn't want is for the application to be a magic box that no one but he understands. A successful implementation of the application at one site should lead to subsequent implementations at other sites.

Prototyping: Pros and Cons

There are a number of primary and ancillary advantages to the prototyping approach. The primary advantage is the saving of scarce systems' and programmers' time. If a working model can be developed on a PC by someone who is close to the problem, let him do so. Then, if the application is providing the benefits that were anticipated, the systems people can study it to see if a division-wide adaptation would be appropriate.

The PC-based prototype can serve as a stopgap measure. If a real problem exists, the PC version can keep the problem from becoming overwhelming until systems can allocate adequate personnel and machine time to address the problem.

Even if the application is a failure, someone had to think through the problem; convince himself, associates, and superiors that this was a problem that a personal computer could be adapted to; and try to do it. This type of activity will familiarize people with the PC and what they are capable and incapable of. This type of practical attempt to use the machines will go further to establish a computer-literate user base than any number of disk-based tutorials or lectures.

Available equipment is more fully utilized, making it easier to convince management that additional equipment is necessary. When the new equipment arrives, opportunities inevitably present themselves.

There are several disadvantages to adopting this type of an approach to systems development. One of the biggest disadvantages is the "Magic Box Syndrome." Unless the application was developed and implemented in the same location, the people who received the application will have no understanding of the project, what it is supposed to do, and what its limitations are. This is why it is so important that enough advance publicity and adequate support is attained before the application is sent to the test site.

If the system that is being prototyped is of division interest and importance, then the data that it contains is probably viewed with this level of importance, too. Generally speaking, PCs are far less secure than the major corporate systems. This means that all of this sensitive information is gathered in one place at each of the locations. Each of these locations has varying levels of security consciousness. The importance of this information makes periodic back-up imperative. As confidence is placed in the system, it may become the sole cache of information that was maintained separately by an assortment of managers. It would be a death blow if the system failed after people came to rely on it. Failing to take adequate security and back-up measures is a prescription for disaster.

Hand in hand with this lack of security is the possibility that there will be a lack of uniform reporting among the various divisional locations. At least in the initial stages, assiduous assimilation and input of the relevant data is left to a person who may feel imposed on. In addition, because the entire system is embodied in the PC, a skilled but unscrupulous user could tamper with the calculations or data input to make his location appear more favorable than it really is.

I believe the advantages outweigh the disadvantages. The possibility of immediately addressing an information processing problem without having to wait until the systems department can work you into its schedule is very exciting. The benefit of increased computer literacy within the organization sets the stage for greater and more significant contributions from all of the players. As users become aware of what can be done, they will demand more from themselves and others. □