Where do we stand in implementing information systems?

by JAMES C. EMERY

University of Pennsylvania
Philadelphia, Pennsylvania

ACHIEVEMENTS OF THE COMPUTER INDUSTRY

By any reasonable yardstick the computer has been a fantastic success. It has grown in one generation from a laboratory curiosity to one of our major industries—perhaps the most vital one in our information-oriented post-industrial society. Raw computational speed has increased during this period by an order of magnitude every four or five years. The cost of computation has dropped to the point that we can employ it lavishly on everyday tasks.

The computer has already brought significant changes in science, engineering, and commerce. Some branches of technology, such as space exploration, would scarcely be possible without the computer. Virtually all scientific fields have been affected in varying degrees. Without the modern computer it would be very difficult to perform the myriad data processing tasks required to run a large organization. Indeed, many of the companies most heavily involved in data processing, such as those in the banking and insurance industries, could simply not exist in their present form without the computer.

PROBLEM IN APPLYING COMPUTERS

Despite these unprecedented achievements, we computer professionals are not universally hailed as heroes. In fact, we have a difficult time staving off our detractors. It would be comforting to think of them as ungrateful wretches who resent the bounty that we have bestowed upon them. Unfortunately, much of our difficulty stems from our own failures to exploit the technology nearly as well as we might. It is all too easy to identify these failures.

Underestimating complexity

Man seems to have a natural bias toward underestimating the complexity of information systems. It is easy to see how this can come about. When we think about a problem we tend to concentrate on the normal combination of circumstances; we often overlook the exceptions. And yet it is the exception that causes much of the work in designing and implementing an information system. It is not uncommon, for example, for over half of the design effort to go into the handling of exceptions that account for a few percent of the transactions entering the system. One approach, of course, is to handle the rare exception outside of the automatic system. But even this requires considerable design effort to detect the exception and control the eventual entry of the corresponding correction.

Just as man seems to have a proclivity to ignore the exception, he also seems to have a bias toward optimism. Estimates are frequently based on the assumption that everything will work out as planned, ignoring the overwhelming evidence to the contrary. Critical difficulties often appear in the process of linking individual subsystems into the overall system. System integration and testing typically account for over half of the total cost of implementing a complex system, but it is the rare estimate that provides this level of funding.

Reinforcing these psychological biases are certain institutional biases toward optimistic estimates. The cost of implementing a large system is very great, but many organizations have not been willing to face up to this fact. More than once I have heard an MIS director say that he could never get authorization approved if he gave honest estimates.

Underestimation of the complexity of a system leads to a number of unfortunate consequences. An obvious one is the cost and schedule overrun that we so often experience. This, in turn, often leads to hasty shortcuts that result in systems that are error-prone, inflexible, and poorly documented.

Design of overly sophisticated systems

Underestimating complexity can lead into the trap of striving for too much technical sophistication. A technician naturally wants to design systems that
move to—or even extend—the frontier of technology. Often overlooked is the cost-effectiveness of marginally useful sophistication.

A common example is an overemphasis on quick response. On-line file updating and retrieval, with response times in the order of a few seconds, can add enormously to the cost, complexity, and technical risk of a system. Alternatives exist that may give nearly the same benefits at much lower cost. For example, skip sequential processing of indexed sequential files permits short batch cycles—perhaps in the order of an hour or less. If this is not short enough, it is possible to provide on-line data retrieval without the much greater complexity that on-line updating requires. There are certainly cases where the extra cost of on-line updating is more than justified by substantial incremental benefits, but these tend to be quite exceptional.

The bias toward sophistication manifests itself in other ways. Designers and programmers often strive too much to achieve machine efficiency. In doing this, they ignore the broader aspects of efficiency that include programming costs, flexibility, and maintainability. We live every day with past mistakes of this sort—for example, the ludicrous situation in which a third generation computer spends its time emulating a second generation computer because earlier systems were programmed in assembly language to gain efficiency. We tend not to make this mistake now; instead, we make errors at a higher level by rejecting, for example, use of a generalized software package because of its “inefficiencies.”

Failure to exploit existing technology

Paradoxically enough, while we strive for sophistication we also ignore well established technology. This is perhaps understandable in a technology that has been expanding so rapidly: it is almost impossible for the practitioner to keep up with his field. Nevertheless, it is painful to see.

A good example of the problem is our failure to include an information retrieval capability within a management information system. The ability to handle ad hoc inquiries has been available for well over a decade. Some companies began to apply such systems in the late 1950s, and proprietary retrieval packages came on the market soon afterwards. Although there is now widespread interest in such systems, they are used much less than they ought to be.

Poor human factors design

One of our greatest failings is the incredibly poor job we often do in serving the human user of our systems. Anyone who has tried to correct a billing error knows the problem. Systems are too unreliable, unforgiving, and uncorrectable. They spew out masses of data that are often poorly identified and displayed in a form that is difficult to comprehend. It is no wonder that users shudder a little when the computer moves in.

IS THERE A BETTER WAY?

Clearly we can do better than we have; the relatively few organizations that have done an outstanding job demonstrate what can be done.

There is no simple prescription for success in this business. Nevertheless, there are certain steps to follow that are probably necessary, but not sufficient, for success. I offer them not as a unique set of guidelines—indeed, they are all well-known—but as a reminder of those things we frequently choose to ignore.

Stay within the state of the art

It is a well-known aphorism in our trade that pioneers are distinguishable by the arrows in their backs. With rare exceptions, it does not pay to develop systems that rely on an extension of the frontier of technology. Even military systems, which have a better justification than most for such a strategy, usually are well advised to stick with proven hardware and software.

Exploit available technology

Staying within the state of the art is not a very serious limitation. The art has advanced to the point that technology does not impose many real constraints. It is perfectly feasible, for example, to design systems that have the following characteristics:

1. Considerable use of shared computer resources. Although the issue of centralized versus decentralized computation is not closed, most advanced systems will find it exceedingly attractive to combine computational loads into a few (often one) large computers. Centralization offers the undeniable advantages of economy of scale, the power of a large facility, load levelling, and the availability of a centralized data base. There will no doubt always exist systems that call for the simplicity and economy of specialization that a small decentralized computer can provide, but for the bulk of our needs we should surely look to relatively centralized facilities.

2. Considerable integration of the data base. The consolidation of files has been going on apace for a number of years. This has resulted in a reduc-
tion of redundancy and greater sharing of data across programs. There still remains, however, the need to link non-contiguous records. This only becomes feasible when one employs a fairly sophisticated data base management system for on-line storage. The capability of linking records has existed for a number of years now—beginning with General Electric’s IDS—but it has yet to be widely applied. This technology has now matured to the point that it should be considered in the design of any new system.

3. Interactive computation. My earlier comment on the overemphasis given to short response time should not be interpreted as a blanket rejection of quick-response systems. Certainly the technology is available to provide reliable on-line response or file updating when the resulting benefits justify the added cost. Our strategy should not be biased in favor of either short or delayed response; rather it should aim at a tailored response that allows each application or user to have the response that best balances costs and benefits. Such a system will very likely provide a wide spectrum of response times—from interactive retrieval and data entry to periodic batch processing.

4. Finally, virtually any well designed system should include some decision making models. The models can vary from analytical optimizing models to man-machine decision systems that rely on elaborate simulation models. It is only when the information system provides direct support of decision making that it really begins to take advantage of the full capabilities of information technology. The most serious issue facing the designers of such systems is the extent to which the models should be embedded within the information system. It is always a difficult task to provide formal, automatic links between transaction processing and decision models. They should therefore be provided only in the case of high-volume and rapidly changing inputs and outputs; in the remaining cases the models can be loosely coupled to the transaction processing through some sort of manual intervention (e.g., smoothing or normalizing transaction data to generate inputs, and reviewing and adjusting decision outputs prior to introducing them into the operational part of the system).

Provide long-term organizational commitment

If there is anything certain about our business it is that the development of successful systems calls for long-term, sustained support. No system is ever really finished; it undergoes continual modification and adaptation over the course of many years. Eventually it may be scrapped and replaced by an entirely new system.

An organization’s commitment to a long-term program should take a number of forms. One requirement is for relative stability in the resources devoted to systems development. Insofar as possible, these expenditures should not be viewed as an easy target for cash savings in time of belt tightening.

Management’s commitment should also take the form of participating in the design of the system and smoothing the way for its introduction. The call for such participation has become a commonplace in our profession, but it is no less true because of that. A well designed system should become an integral part of the management process of an organization, and it is hard to see how this can come about without continual support by all levels of management. Many important management policy decisions are embedded within the MIS design, and so managers must participate to the extent required for them to resolve these issues.

A long-term commitment implies an explicit program to develop systems personnel. The need is especially critical at the management level within the systems group itself. The philosophy that any good manager can head a systems group has surely been discredited by now. The body of knowledge within the field has grown to the point that this knowledge cannot be quickly or casually acquired; the development of real professionals requires a long-term program. We should be no less reluctant to turn over systems development to the untrained and unseasoned manager than we are to trust the removal of an appendix to a barber.

An organization’s commitment is also measured by the calibre of personnel they assign to the MIS function. Little success can be expected if the function is considered a dumping ground for misfits, if it does not get its share of the organization’s fair haired boys, or if it does not attract outstanding computer professionals. Without at least a sprinkling of really top-flight designers, analysts, and programmers, the organization would be well advised to scale down its aspiration level to meet only bare requirements.

Emphasize users

Everyone talks about the need to serve users, but we nevertheless continue to make many of the same mistakes. Evidently we must take much more formal and explicit steps to deal with this problem.

The use of steering committees is one way to get user
participation. They can certainly help in making re-
source allocation decisions and in resolving policy
questions. They are equally applicable at all levels in
the organization—a high-level committee to deal with
broad, long term issues, for example, and lower-level
committees to deal with operational subsystems. The
form of the committees and their tenure may vary
considerably, but certainly some relatively formal
means is needed as a forum for dealing with issues that
cross existing organizational boundaries.

System groups should include staff members who are
specifically assigned responsibility for human factors
design. Included within their scope of activity should
be such matters as the design of hard copy forms and
CRT display formats, procedures connected with the
information system (including error handling proce-
dures), specification of the operating characteristics of
terminals, and concern for the ability of the system to
adapt to changing user needs. These responsibilities
tend to be scattered about within most groups, rather
than dealt with as a whole.

A formal user sign-off procedure should exist so that
users have an explicit opportunity to get their views
incorporated into the MIS design. Such sign-off should
be required at all major stages of the implementation—
the feasibility study, functional design, detailed design,
testing procedures, and final operational acceptance.

It might be a good idea for each organization to have
an ombudsman to look out for users’ interests. As in-
formation systems become more and more pervasive
throughout the organization, we stand some chance of
being swallowed up by them unless someone is around
who can pull the plug in an emergency. To be effective,
the ombudsman should remain independent of the
MIS group, have the authority to investigate all matters
connected with the information system, and have the
right to voice his opinions at the highest levels of man-
gement. One would probably not find enthusiastic
support for such a position among systems directors,
but it nevertheless may provide a very useful safeguard.

Devote more effort to project management

The implementation of a large MIS has itself become
a formidable management task. It may require the
efforts of over a hundred persons extending over the
span of several years. The various subsystems tend to
interact closely with one another, and therefore call for
a high degree of coordination among the project teams.

An activity as complex as this requires more atten-
tion to its management than normally given. Tech-
niques exist for dealing with the problem—network
scheduling and resource allocation models, documenta-
tion and configuration management procedures, simu-
lation packages for analyzing alternative designs, and
all the rest. Unfortunately, however, most projects do
not formalize project management to the extent neces-
sary. The direct cost of good management is quite
high, but the indirect cost of poor management is
staggering.

Centralization of systems management

It is hazardous to generalize too much about the
proper degree of centralization of systems activities.
Organizations thoroughly committed to decentralized
management find it hard to treat the systems activity
as a centralized exception. Centralization can also in-
crease problems of keeping the system responsive to
users’ needs.

Nevertheless, considerable benefits can come from a
certain amount of centralization. The benefits basically
stem from the economies of shared use of hardware,
software, and data.

For most organizations it does not make much sense
to maintain more than one group concerned with the
development of standards for programming, docu-
mentation, languages, data base design, and project
evaluation. Vendor hardware and software evaluation
should likewise be centralized in most cases. Design of
common application programs might even be central-
ized. Clearly we are already seeing more and more
centralized computer facilities.

Centralization of this sort does not imply isolation
from users. Specialized applications can—and probably
should—be handled on a decentralized basis. A remote
terminal hooked to a centralized computer should be
viewed logically by decentralized managers as an inde-
pendent resource with the capabilities of a large machine
at a fraction of its full cost. To be sure, any centraliza-
tion imposes some constraints on the organization’s
subunits, but when intelligently applied the constraints
still can leave ample room to meet users’ requirements.
After all, we all must live within certain constraints,
whether imposed internally or externally by govern-
ments or vendors.

Incremental development

A comprehensive system takes many years to de-
velop. No one can foresee how technology and users’
requirements will evolve over this length of time. It is
therefore absolutely essential that systems be designed
in a way that permits continual modification and
extension.
Where do we stand in Implementing Information Systems?

A number of steps can be taken to achieve a much higher degree of flexibility than most systems offer. Perhaps the most important requirement is a master plan to guide the development of separate subsystems. Such a plan defines the subsystems that will comprise the eventual overall system, and establishes the major interfaces among these subsystems. It is only when such a plan has been developed that one can have any assurance that all of the subsystems will mesh together.

The master plan need not go into great detail. Its purpose is to establish constraints within which each subsystem can be developed more or less independently. In principle, one should limit the detail to just that level required to define the interfaces among subsystems. Detailed work beyond this level can wait until the actual implementation of the separate subsystems.

Insofar as possible, the subsystems should be implemented in a sequence that provides early interim benefits. Certain technological constraints may partially limit the choice of sequence—for example, some generalized software may be required before application programs can be written, and a forecasting program cannot be implemented until the order entry subsystem from which it gets its inputs has first been developed. Other factors, such as the particular skills available and political considerations, may play a part in choosing the implementation sequence. The basic strategy should be, however, to implement each subsystem in a way that provides interim benefits that justify development costs, while at the same time leading toward the eventual accomplishment of the master plan.

The plan should evolve along with the system; it is not meant to impose inflexible constraints on the design. Changes will inevitably occur as technology advances, the organization’s needs change, and the system matures. The changes should, of course, be controlled and consideration given to trade-offs and the need for coordination among subsystems.

In addition to the development of an evolutionary master plan, other well-known techniques exist for increasing the flexibility of a subsystem. These include use of higher-level languages, modularity, data and program independence, avoidance of unnecessary program complexity, and careful documentation. These techniques typically carry a short-term price, but they are vital for long-term success.

CAN SUCCESS BE REPLICATED?

We know that real success can be achieved, but we also know that it is relatively rare. This is so because success requires the rare combination of exceptional management, sustained support, and top-flight technical personnel. Without these ingredients, results are likely to be marginal at best and disastrous at worst.

Unfortunately, there is not enough talent to insure that each organization has an adequate supply. The great challenge we face is to develop a technology that does not require genius to implement. Conversion of a field from an art to a routine science is the chief sign of its maturity. To achieve this in our field we must devote a great deal more effort to gaining a better theoretical understanding of complex systems, planning and control of large organizations, and the behavioral aspects of formal information systems.