Welch, James A., and P. Ranganath Nayak. "Strategic Sourcing: A Progressive Approach to the Make-Or-Buy Decision". *Academy of Management Executive*, Vol. 6, no. 1 (1992), pp. 23–31.

Yates, John C., and Henry W. Jones (*III*), 'Computer Acquisitions: The Role of Due Diligence', *Computer Negotiations Report*, Vol. 8, no. 9, (1985), pp 1–4.

■ Two lawyers give advice on the importance of knowing the vendor before signing a contract (including the vendor's litigation record) and suggest five ways of obtaining information about the vendor. The journal itself is a valuable source for articles on acquisitions and contracts but is difficult to obtain.

Bsc-III SAD
Unit III & IV

BSc-II nd sem
Unit III

16

Organizational Adjustments, Testing, and Conversion

There is nothing permanent except change.
Heraclitus (540–475? BC)

A new information system affects the manner in which tasks are performed in an organization. Some jobs are eliminated. Others are created. Many employees find that their job descriptions are altered which means that new skills and new methods of operation must be learned.

During systems development, manpower needs for the system are identified. During systems implementation the workforce is expanded through hiring (if needed). In addition, data processing personnel and end-users are trained to interact with the system and changes may be made to the organizational structure of the firm. Some departments may grow in size while others retrench. Some operational managers may be given a broader span of control because of the new information system while others see their 'empires' shrink. To implement such organizational adjustments often taxes the skill of corporate management since employees typically resist change.

With the acquisition of hardware and preparation of software, testing begins. The test process starts at the component level and continues until the system as a whole meets testing standards set by end-users, management, and the development team. Conversion, the switchover to the new system and dismantling of the old, follows. A final activity before the development team is disbanded is evaluation of the systems development process. All of these activities are discussed in this chapter.

TASK IDENTIFICATION

The specification of manpower needs for an information system under development takes place during systems design. Each task or operation associated with the new system is listed by analysts at that time.

If operation of the new system will be the responsibility of a data processing personnel, this list will be forwarded to the manager of the systems department who will distribute the workload among staff or, if necessary, hire additional help. The list can be used to structure jobs, to plan changes to allow for job enlargement and job enrichment, to write job descriptions, and to ensure that all tasks are allocated.

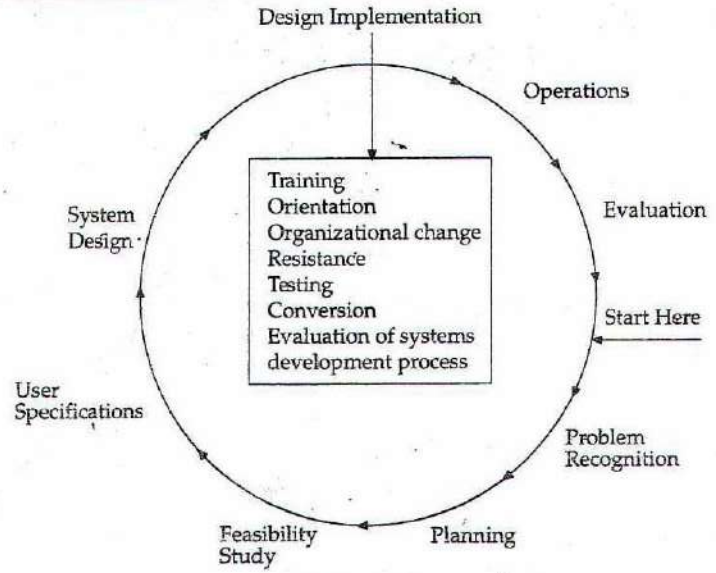


Fig. 16.1

If the new system under development will be under the jurisdiction of end-users, the *task list* will guide analysts in the preparation of end-user manuals and training materials. It also helps those who are responsible for formal training programs to identify what skills to teach.

TRAINING

Questions that must be addressed by the development team with regard to *training* are:

- What procedures or skills does the new system require?
- Who needs to be trained?
- What type of training is appropriate?
- Who should do the training?
- When should training take place?

While some of the employees who need training will have technical expertise and computing experience, others will be computer novices. Some employees need to learn how to input data in the new system, some how to operate the equipment, and some how to generate and interpret output. For these reasons, a number of homogeneous training groups may have to be established with a unique training agenda for each group. This may be more costly in time and teaching resources than a single training program for everyone, but training will be far more effective.

A *training matrix* similar to the one in Fig. 16.2 can be used to help organize training sessions. The vertical axis identifies the training modules—the horizontal axis, the people to be trained, how training will be performed, who will do the training. Time is shown in the matrix cells. The training itself may consist of on-the-job training, formal lectures and seminars, workshops, briefings, slide shows, manuals, role playing, or computer-aided instruction. The latter has an advantage over formal classwork because learning is self-paced and scheduling can be planned for employee convenience. Many vendors contribute to training by providing computer-based training materials for their products. Other training options are to send employees to commercial training institutes or schools run by manufacturers, and to release time for classwork at local colleges or universities.

What to be trained for? / Who is to be trained?	*Marketing Management*	*Mr. Jay Clarke*	*Joe Smith Supervisor*	*Clerks*	*J. Amiu*	*Who is to be trained? / How is the training to be presented and by whom?*
Overview • Technical	1/10 1–3 pm	1/10 1–3 pm	1/10 1–3 pm	1/10 3–5 pm	1/10 1–3 pm	Lecture by A Banerjee
• Organizational	1/10 11 am	1/10 11 am	1/10 11 am	1/10 1 pm	1/10 11 am	Lecture by R. Gupta
Output	1/11	1/11	1/11	1/12	1/12	Lecture— M. Agarwal
	10–12 noon	10–12 noon	10–12 noon	1–4 pm	4–5 pm	Lecture— M. Agarwal
Procedures • Overviews	1/12 9–10 am	1/12 9–10 am	1/12 9–10 am			Seminar by M. Agarwal
• Details			1/13 1–3 pm	1/13 1–3 pm	1/13 1–3 pm	Seminar by M. Agarwal
Equipment	1/17 8–10 am	1/14 8–10 am	1/14 10–1 noon	1/19 all day	1/21 1–3 pm	Hands-on Supervised by: to be announced

Fig 16.2 Training Matrix

Figure 16.3 is one example of the way in which training can be sequenced. Note that each type of training is based on design specifications. But not all firms follow this model. For example, a small organization may train a single individual in input/output procedures and equipment operation so that consecutive training sessions rather than parallel sessions would be required. Whoever does the training will also affect schedules. In some companies, analysts on the development team are assigned training responsibility. Large firms often have an education department

Employees at operational levels, such as assembly line workers and technicians, may be as disturbed by disruption of the *status quo as* professionals and managerial personnel. People are creatures of habit and become upset when familiar work routines are altered.

The main reason managers resist information systems is that computers alter the decision making process. In a computerized environment, decisions are largely based on data provided by the computer supplemented by human judgment and experience. This requires a different type of conceptual thinking than intuitive decision making common in the past. Some managers feel hemmed in by information technology, claiming that computer systems centralize important decision making. Some managers resent having daily goals defined, the action to achieve these goals specified, and performance evaluated by what they perceive as in impersonal machine. A computer system may also provide data that managers prefer to suppress.

Even data processing personnel may resist the implementation of a computer system under development. Like non-technical workers, they fear unemployment as advances in computer technology foster change in their work assignments. Many computer specialists view end-user computing and distributed data processing, which transfer responsibility to users for the operation and administration of computing resources, with alarm. They believe that they lose status and power as a result because end-users no longer require their services as intermediaries to interact with computer systems.

Reasons for resistance to change are summarized in Fig. 16.4. Although resistance to the implementation of a new information system can impede conversion and reduce the systems effectiveness once it is operational, it should be acknowledged that not all resistance is bad. If management examines the objections of employees and improves the proposed system by listening to constructive criticism, resistance can serve a useful purpose. Perhaps technicians have not paid enough attention to

Level Primarily Affected / *Reason for Resistance*	*Operating Personnel*	*Operating Management*	*Middle Management*	*Top Management*
Loss of status	x	x		
Economic insecurity	x	x	x	
Interpersonal relationships altered	x	x	x	x
Change in job content	x	x	x	
Change in decision-making approach		x	x	x
Loss of power	x	x	x	
Uncertainty/unfamiliarity/misinformation	x	x	x	x

Fig. 16.4 Reasons for Resistance and Effects of New Technology on Employees

human needs in designing the work environment. Perhaps the emphasis has been on technology, ignoring human–machine interface on which success of the new system depends. A careful assessment of employee objections may help avert a costly flop. Those who assume that all resistance is the grumbling of malcontents do their firm a disservice.

Planning To Implement Change

Often systems analysts act as *change agents*. But firms that have little experience in the management of change may hire a consultant as a change agent. The job of a change agent is to recommend policies that will smooth transition to new technology and suggest ways to alter the behaviour and attitudes of employees resisting change.

Skilled change agents know that employees must be made aware of a proposed change and have time to grapple with what the change means to them. Initial confusion or a lack of understanding of the ramifications of change should not be misinterpreted as resistance. To avoid the heated, emotion-charged environment that accompanies sudden change, employees should be told early in a new system's development the reasons why new technology is being introduced and what will happen as a result. They will then have time to reflect on the change and be able to evaluate the pros and cons of the project with greater objectivity. Given a period of adjustment, employees have time to learn new job skills or time to seek alternate employment.

Experienced change agents also recognize that an individual's assessment of new technology may be mixed, fluctuating between positive and negative emotional and intellectual judgements. At one moment there may be hope that work in a computerized environment will lead to professional advancement—at another, deep-felt anxiety of job displacement or unemployment. If the change agent can accentuate positive reactions to the new technology, commitment to change will follow.

There is a cycle in change attitudes, as illustrated in Fig. 16.5, that the change agent hopes to influence. Each person's emotional response to change is preconditioned by his or her background, education, and experience with change in the past (*Box 1*). Emotional responses lead to attitudes (*Box 3*) that affect whether the impact of innovation at work is perceived as beneficial or as a threat (*Box 4*). This perception, in turn, influences whether change will be accepted or opposed (*Box 5*) and the degree of willing participation in the implementation of the change (*Box 6*). This experience becomes part of the employee's background (*Box 1*), conditioning future reaction to change.

The change agent hopes to interject new experiences or knowledge into this cycle and reinforce positive responses to change by handling change in an equitable and humane manner (*Box 2*). As a result, it is hope that the employee's attitude toward change will be favourably altered and that resistance to change in the future will be diminished.

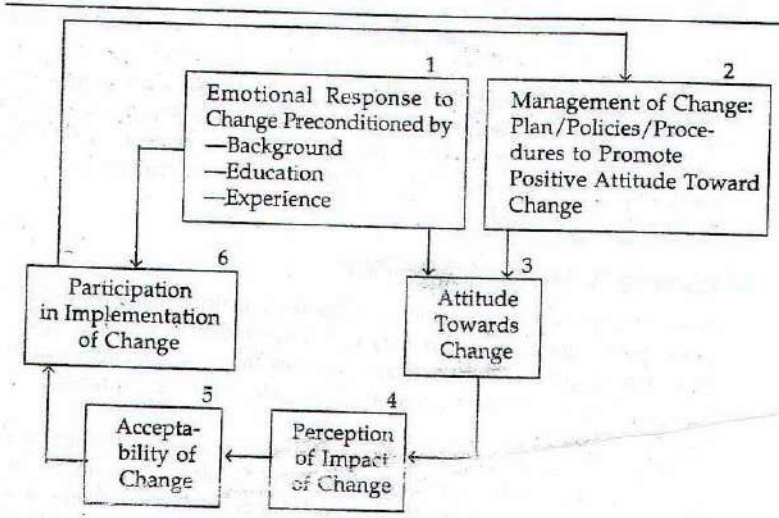


Fig. 16.5 Cycle In Attitude Towards Change

Change Strategies

Training and orientation, described earlier in this chapter, are a key to favourable attitudes towards change.

Other ways for management to smooth transition to new information technology are to:

- Assign responsibility for change to upper level managers who possess the organizational power to legitimize change.
- Identify individuals in the organization who must learn new behaviours, skills, or knowledge because of the change and schedule training for them.
- Involve resisters in the development of new systems by giving them an active role in identifying problems and planning solutions.
- Open lines of communication between employees and management. For example, provide forums where employees can voice their concerns about the new technology.
- Avoid secrecy about the new system. Publicize information regarding system changes.
- Pace conversion to allow a readjustment period to the new system.
- Implement new systems in modules.
- Alter job titles to reflect increased responsibility.
- Reward ideas that will improve throughput.
- Document standards so that new procedures are easy to learn and reference.
- Clearly establish in advance the demarcations of authority that will exist following changeover.
- Upgrade the work environment following change, incorporating recommendations of human factors studies.

- Show sympathy and be receptive to complaints following conversion.
- Give job counselling.
- Arrange job transfers.
- Call a hiring freeze until all displaced personnel are reassigned.
- Provide separation pay.
- Organize group therapy.
- Initiate morale-boosting activities, such as a company newsletter and social events.

The key to success of these strategies lies in management's ability to demonstrate support for employees adversely affected by the change. Management should also exhibit patience and understanding when the disruption and dislocation of change produces anxiety and tension even among those not directly involved with the new system. It is largely a skill in handling interpersonal relationships that determines whether a firm can absorb technological advances in the field of computing.

TESTING

Testing is a quality control measure. The information system under development, or parts of the system, are exercised to see if the results satisfy performance specifications and match anticipated results. If not, the reason(s) are traced, modifications are made, and the system is retested—a cycle that is repeated until testers are satisfied with quality.

The underlying premise of testing is that correct results can be predicted and that the validity of a system can be ascertained by comparing results generated during a test against predicted results. Key elements of the system are identified, then duplicated and checked in a production environment that matches the environment in which the system will operate once conversion takes place. Test cases are selected and test data are generated to simulate live, full-sized files and transaction volume. If planning lacks thoroughness and organization so that key elements of the system are not checked at some point, the system may run successfully during the test cycle yet fail when placed in operation. The problem is that a complete definition of test cases is virtually impossible and that exhaustive testing can pass the limits of practicality.

What elements must be checked?

- The hardware on which the new system will operate.
- The operating system that will be used.
- The assembler or compiler that will produce the object code.
- The data to be processed.
- Applications software.
- Data entry methods.
- Operating procedures.
- Output interpretation.

In general, systems are checked for reliability, effectiveness, file integrity,

audit trail, execution validity and proficiency, ability to handle expected workload and generate output within time constraints, service level, backup and recovery, security, documentation, and compliance with standards. Although testing generally precedes conversion, testing may take place at different times in the development cycle; for example, when training is completed, hardware installed, procedures designed, software packages purchased, or in-house programs written. However, total system testing must wait until all components of the system are on hand and conversion to the new system is ready to begin.

Too often testing takes place without first setting standards or formal procedures to insure the quality of the tests themselves. As a result, errors pass undetected that later surface and require costly maintenance. From a monetary standpoint, careful planning of the test phase makes sense—it is much less expensive to correct errors during systems development than to make corrections once systems are operational. The fact that testing may consume a large share of development costs, as illustrated in Fig. 16.6, is an additional financial reason to plan tests with care.

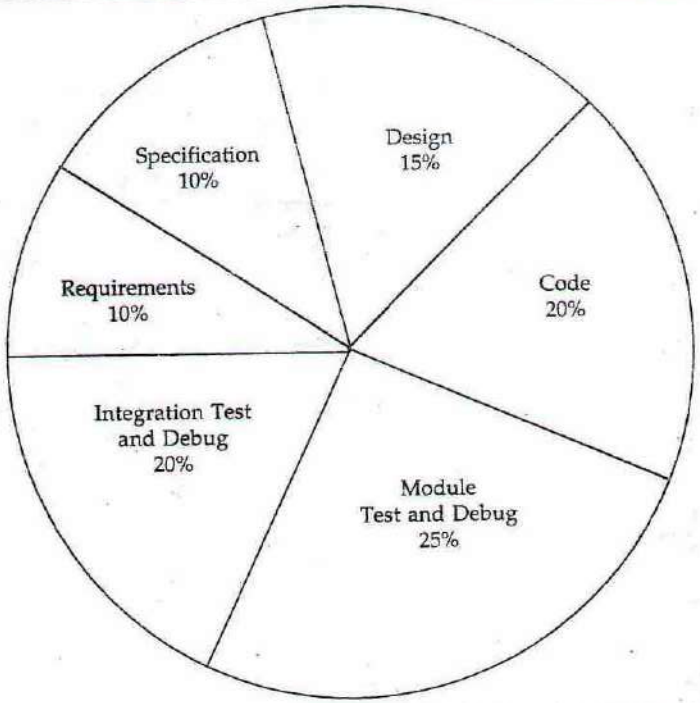


Fig. 16.6 Breakdown of Development Costs for a Typical System

Well-planned tests have four phases: preparation for testing, execution, evaluation of test results, and the final system test.

Preparation for Testing

Tests *standards* are established in this first phase of testing. For example, reliability testing standards that state the allowable occurrence of error in processing are formulated. So are performance standards (specification of response time, throughput, and staff and equipment efficiency) and load standards for peak periods and future expansion.

Then *test cases* are enumerated. Since a development team cannot conduct every conceivable test, priorities for testing should be established. One development team may emphasize the testing of typical inputs. Another may focus on boundary value cases. Some teams look for test cases least likely to draw the attention of analysts and programmers responsible for systems design and coding. Others want to ensure that old, familiar capabilities satisfy users before testing new features that may be unused for months following conversion. Variations in test priorities are to be expected among development teams.

Next, *test data* is collected. The team must decide how test data will be generated, what data to use, and the amount of data required. In many cases historical data is available for testing. For example, in converting from a manual to a computer system, the new system can be tested against the results of the old, using data collected in the past. However, in projects performing functions never done before, test data must be created. This should be done by the user (who can anticipate the conceptual problems and exceptions that may arise once the system is installed) in cooperation with analysts (whose experience makes it possible to anticipate operational problems).

What data to use and how much to use is a problem of sampling. A well-designed sample produces results as significant as those from full-sized files. Unfortunately, few people working with systems have training or applied experience in empirical experimental techniques. It may be necessary for the development team to add consultants knowledgeable in experimental design for preparation of test data. Consultants may also help predict what correct output should be and help develop acceptability standards.

Of course, small information systems do not require as elaborate testing as large, complex systems. But to eliminate the test phase, even for simple applications, is foolhardy.

Test Execution

A number of automated *testing tools* are available to the development team that help reduce costs and increase the thoroughness and accuracy of testing procedures. These are described in Box 16.1. The testing itself is done at four levels—component testing, function testing, subsystem testing, and system testing.

BOX 16.1

Testing Tools

A *program execution monitor* can be used to create a record of the program statements that are executed during a test run. The monitor can also produce reports that summarize execution characteristics. It can evaluate the logical completeness of a given set of data, isolate unexercised code, and display the activity of statements of a certain type, such as all IF statements.

A *test data generator* is a program that can create a test data base from information supplied by the user, such as definition of the data base structure, sequence in which records appear, record characteristics, field characteristics, field formats, and range of permissible values. The file created may be used at first to test normal functions on a minimal set of transactions. The test data generator may then be used for exception testing of invalid transactions or to generate a much larger test data set for high volume testing. Of course, the quality of the data set depends on the completeness and accuracy with which test cases are defined. The development team must still decide what to test, how testing should proceed, and what results to expect. Only the mechanics of file creation are automated.

A *terminal input simulator* can be used to test on-line systems before all terminal have been installed. It imitates the way in which a multiple-terminal environment operates, presenting test transactions that simulate volumes and types of transactions that can be expected. *File conversion programs* can be used to create test files from the old system. Sometimes the accuracy of new systems is checked by *parallel testing*: the comparison of results of the old and the new system programmed to accomplish the same task. Sometimes software is checked by *redundant programming* to see if a discrepancy in results will occur, indicating an error in one of the systems.

Component testing is checking the parts of the system: for example, the operation of a piece of equipment, the performance of an individual employee, or the effectiveness and correctness of a form, procedure, or program. It is estimated that 50 per cent of testing is spent testing program parts to see that individual program statements, subroutines, compound statements, or subpaths through a program are correct. There may be errors in design logic or in the writing of program code. To find one bug per hundred statements is not uncommon.

Testing usually begins with component testing because it is easier to identify errors and problems at this level than it is to isolate problems when testing integrated components. Also, the earlier a problem is detected, the less effort is required to correct it.

Function testing is at a higher level of aggregation—checking the performance of related components that work together in a functional subsystem. This level of testing should be done after components of the function have been tested individually and found satisfactory.

Subsystem testing evaluates how related functions operate. For example, in testing an input subsystem, input procedures, input preparation, validation checking,

and procedures for correcting errors in input would be assessed. Testing at this level checks the interrelationships of functions tested individually earlier.

System testing, checking overall results (that is, the performance of all subsystems as a unit) is the fourth test phase.

Perhaps a practical example will help make the concept of testing levels meaningful. Figure 16.7 shows how testing of a financial system is subdivided into subsystems. The figure further illustrates how a sample subsystem, accounting, is subdivided into functions for testing. In addition, the components of one of these functions, payroll, are listed. Testing would begin at the component level of this hierarchy, continue to higher levels of aggregation, and end with a total systems check.

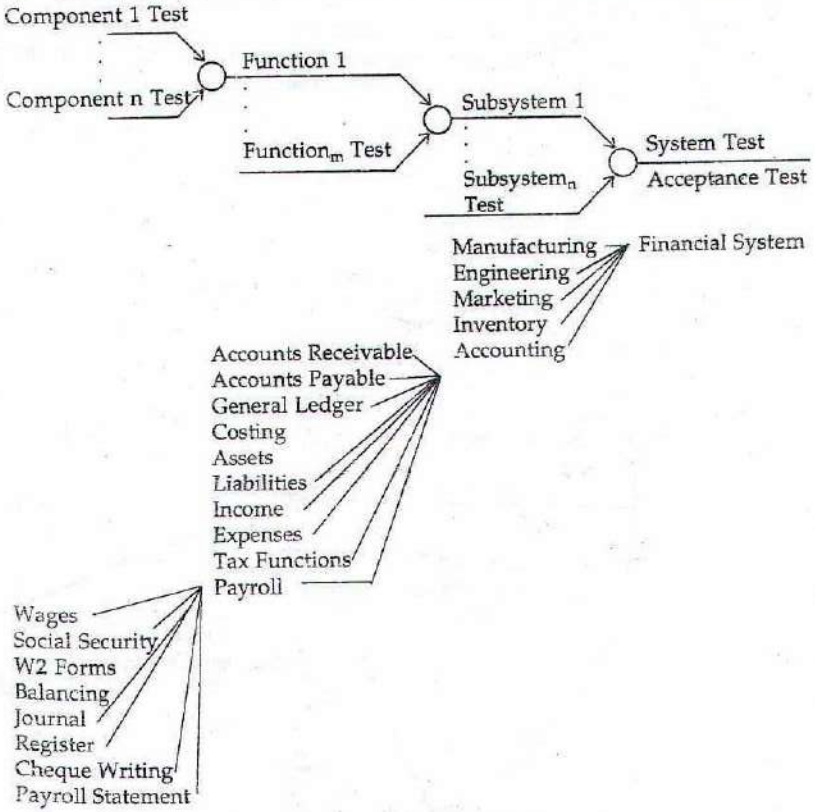


Fig. 16.7 Levels of Testing

Some tests are run in an artificial testing environment. For example, vendors who design systems for clients may run preliminary tests in-house on their equip-

course, the project manager of the development team is ultimately responsible for the development of the new information system. Although this manager will not personally conduct tests and evaluate results, he or she should set test standards and spot-check testing to see that standards are followed. The project manager should require documentation of test results and procedures. This documentation can be used to verify that test results are reproducible.

Systems analysts on the development team plan the tests and coordinate the testing effort. They see that programmers test all unit software before turning over the modules for system testing. Analysts also ensure that vendors check their hardware to make sure it operates in the production environment and meets contract specifications. For large systems, computer operations personnel provide the manpower and machine time to execute system tests under the direction of analysts on the development team. When programs involve funds or accounting procedures, an internal auditor is generally involved in the test function. When the new information system will be operated and managed by end-users, they will want to conduct final acceptance tests. End-user management helps set testing standards and develop a detailed test plan with analysts on the development team. Corporate management might assign personnel to participate in the tests and review results.

Sometimes an outside firm that has not participated in the development of the system is contracted to conduct system testing. This helps to ensure objectivity in testing procedures and evaluation of results.

CONVERSION

Implementation concludes with the cut-over or *conversion* to the new information system. Outdated forms and files are withdrawn and equipment no longer required is removed.

A number of conversion options are available, each with advantages and disadvantages. The method chosen by a development team has an impact on implementation costs, on the length of time required for conversion, and on the morale of end-users and systems personnel during the conversion process.

Cold Turkey Method

As the name implies, *cold turkey conversion* consists of complete abandonment of the existing system and immediate use of the new information system without time to plan for the switchover. Usually complete systems collapse or a disaster such as a fire that destroys tapes and hardware leads to this type of conversion. This conversion method places enormous pressures on the development team because problems inevitably arise that have not been anticipated and there may be no backup system to take over the workload while solutions are being studied.

On the other hand, everyone recognizes the crisis and tries to make the system work in the shortest time period possible. Instead of resistance to changed work patterns, cooperation is the norm. Both end-users and systems personnel become

problem-solvers, not problem-creators, with unity of purpose uncharacteristic of other conversion modes.

Parallel Conversion

During *parallel conversion* both the old and the new systems are run concurrently for a period of time. Users are able to validate the output generated by the new system while still placing their trust in the old. They have time to become comfortable with the new system and time to gain confidence in its ability to meet system specifications before the old system is discontinued.

The disadvantage is that running two systems in parallel doubles the workload, adding stress to the conversion process for end-users and systems personnel if manpower is stretched thin. Redundancy may require costly overtime and lead to tiredness, irritability, and frayed nerves that add friction to the transition period. There is also the temptation to return to the old familiar system if problems arise.

The Pilot Method

The *Pilot conversion method* means that a small group of end-users is selected to implement the new system while other end-users continue to rely on the old. Over a period of time, more and more end-user groups convert to the new system until old methods of operation are completely phased out.

The principle advantage of this method of conversion is that it gives systems personnel experience with a controlled volume of activity in a 'live' environment before total conversion takes place. The development team learns how to manage the transition period and identifies problem areas that call for additional debugging, revised procedures, or special training of end-users. Technical problems that arise are on a small scale, affecting only a few end-users, so that disruption in the workplace while analysts look for solutions is minimal.

There are disadvantages, however. The pilot method is lengthy, extending the conversion process over a long period of time. In addition, two systems must be supported concurrently, which adds to information processing costs.

The Modular Method

Modular conversion, sometimes called piecemeal conversion, is similar to the pilot method—both phase in the new system while the old is still in operation. During modular conversion, however, individual components (modules or subsystems) of the system are placed in operation one at a time. For example, a component in a financial system might be a general ledger module.

Modular conversion is common for large and complex information systems. When users have time to learn components over a period of time, there is less confusion, less disruption, and consequently less resistance in the workplace than might otherwise occur. This mode of conversion can be supported by a limited number of technical personnel, another advantage. But not every information system is modular in design so this conversion method is not always practical.

Another disadvantage is that modular conversion can extend over a longer time period than desired and add to conversion costs.

Sequential Conversion

With the *sequential approach*, a complete switchover from the old to the new system is made on a given date . This mode of conversion requires careful planning. Data processing staff and end-users must be well trained in the operation of the new system and patient when problems arise that require system adjustments following conversion.

One advantage is that conversion costs are low since there is no delay in phasing out the old system. Users who might drag their feet and resist the new technology have no choice but to support the new system. Many development teams believe that confidence in a new system only develops when it finally stands on its own. Should the system fail, however, the consequences are far greater than conversion with the old system running as backup.

Conversion Period Length

Generally the development team continues to monitor systems performance for a period of time following conversion to fine-tune the system, should problems arise or bugs appear. The length of this period varies according to the complexity of the project, the need for adjustment, the number of projects queuing for development that require the attention of analysts on the team, and the satisfaction of end-users. It is generally a corporate management decision when responsibility for the new system is transferred to systems maintenance personnel. Usually a formal sign-off takes place in which end-users officially accept the system. This releases the development team from further obligation to the system.

EVALUATION OF THE SYSTEMS DEVELOPMENT PROCESS

The final activity that takes place before the development team is dispersed is *evaluation* of the systems development process. At this time mistakes should be identified and analyzed to see why they were made and how they could have been avoided. The purpose of such an analysis is not to exchange accusations and recriminations, but to document problems and their solutions so that future development teams can learn from past experience and not make similar mistakes. For example, a critical review of the need for recycling during the development process, the identification of liaison and communication problems that the team experienced, and a listing of reasons for schedule slippage should lead to improved procedures and approaches to development in subsequent development projects.

Evaluation of the new system itself is the subject of the next chapter, Chapter 17.

SUMMING UP

Implementation of systems design consists of hardware acquisition and installation, in-house programming or the acquisition of software from an external source, and data base preparation described in Chapters 14 and 15. In addition, training takes place, a topic described in this chapter. Because training may take a long period of time, it must be started early. In general, training begins as soon as manpower requirements for the new system have been identified.

Every new information system results in changes in job assignments in user departments and may even involve changes in the organizational structure of the firm. Usually corporate management makes such changes, but analysts on the development team, acting as change agents, may suggest ways to create an environment receptive to the new technology.

Testing is an implementation phase in which the performance of the new system is measured against desired and expected performance. Following evaluation of test results, the system may be recycled to an earlier phase of development in order to correct errors uncovered during the tests. Recycling is usually done under great pressure of time, and therefore requires a heavier workload than usual on the part of development team.

Conversion, like testing, is a period of organizational strain. Normal service is unavoidably affected. The stress of conversion can be minimized if the problems of conversion are understood and anticipated, and if an appropriate conversion method is selected. Conversion options include the cold turkey method, parallel conversion, the pilot method, the modular method, sequential conversion, or some combination of these methods. For example, pilot and phased conversion could be done in parallel with existing operations, or implemented sequentially. Organizational breakdowns are most likely to occur during long parallel runs.

A final duty of the development team is to present an evaluation report of the development process to management. This report should include mistakes made during development and insights into the development process that might aid future teams in the development of other information systems.

CASE 36

SELF-SERVICE GROCERY CHECKOUT

Self-service in supermarkets is now moving to the checkout counter. The Kroger Company, a grocery chain, is installing an automated system called Service Plus in trial stores. Checkout lanes have no checkout clerk. Instead, customers calculate their own bills by passing grocery items across a scanner that reads the price code on each product. Customers can be assured that all items have been accurately accounted for by reading a monitor that displays the name and price of each item that passes the scanner. Payment for the groceries is made at a cashier station.

Questions

1. How would you test this system? Design a test plan for all levels of testing, including component testing, functional testing, subsystem testing, system testing, and acceptance testing.

2. What type of organizational changes will be required in supermarkets if Service Plus (or a similar system) becomes widely used?
3. Will self-service checkout be resisted? By whom? Why?
4. Describe ways that resistance to self-service checkout might be dissipated?
5. Who are the end-users of Service Plus? How might they be trained in use of the system?
6. Suppose you were a store manager converting to Service Plus. What conversion method would you favour? Justify your position.

CASE 37

EXPERT SYSTEM FOR AMERICAN EXPRESS

An expert system called 'Authorizer's Assistant' is being launched by American Express to weed out bad credit risks. When a customer requests credit, the system draws upon data in company files about that customer, such as outstanding charges, payment history, and buying habits, to ascertain whether to grant credit or not. The recommendation of the system is based on authorization rules written by knowledge engineers who developed the software. They interviewed five best authorizers from American Express in order to form the system's knowledge base. A pilot version of Authorizer's Assistant, which debuted in November 1986, had 800 rules.

'Live' authorizers who use the software can override a decision to grant credit if they think the system has made a wrong call. The final credit decision is still a human one. Nevertheless, it is expected that the advice of the system will be taken in most cases since expert authorizers are impressed with the system's authenticity.

Besides helping to cut losses, it is hoped that this expert system will shorten the total handling time of each credit transaction by as much as 20 to 30 per cent.

Source: Alan Alper, 'Expert System vs Credit Fraud', *Computerworld*, Vol. 21, no. 15, (April 13, 1987), pp 25–30.

Questions

1. What impact does this expert system have on the organizational structure at American Express?
2. Will the fact that expert authorizers at American Express contributed to the development of the system affect how the system is received at American Express? Explain your answer.
3. American Express is the first charge card company to utilize artificial intelligence in credit authorization applications? Do you think that other companies will follow this lead? Why or why not?
4. Who should be responsible for testing this system? What types of tests should be conducted? At what level in American Express should final approval of the system take place?
5. Give reasons for and against the following conversion methods in the implementation of this expert system:
   (a) Cold turkey method
   (b) Parallel conversion
   (c) Pilot method
   (d) Modular method
   (e) Sequential conversion

CASE 38

ELECTRONIC TRADING SYSTEM AT THE LONDON STOCK EXCHANGE FAILS

A 1986 crash at the London Stock Exchange was not caused by the falling price of stocks. It was the crash of an electronic trading system, consisting of five networked IBM PC ATs with an additional AT network controller, that was designed to handle options. (An option gives the holder the right to deal in a share at a future date at today's prices.) The failure of the system caused the Exchange to halt options trading for a day in order to clear up the backlog left by the crash.

The options system was designed to match buyers and sellers, an activity previously carried out using a mainframe-based batch system at the London Options Clearing House. The crash occurred on the first day that the new system was in operation. Its cause? A hardware fault in one of the terminals that locked up the network. Many brokers blamed the failure on inadequate testing: conversion to the new system took place after only three days of a parallel run. Another criticism heard at the exchange was that the system lacked a backup facility, although one had originally been specified.

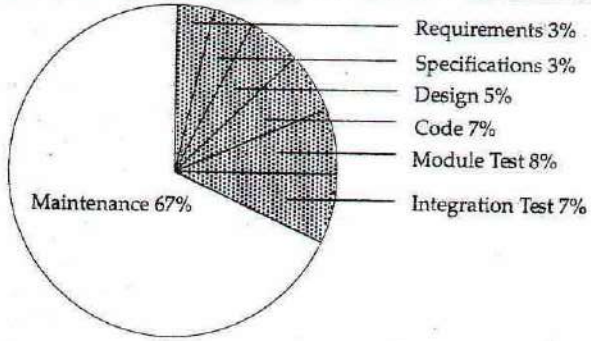
An official inquiry has been initiated to find out the reasons why the system collapsed and what safeguards are needed to prevent future failures. The cost of the closure is also being assessed.

Questions

1. Can failure of this system be attributed to poor systems design? Explain your answer.
2. Design testing specifications for this system that might have prevented this 'crash'.
3. Why do you think no backup facility was implemented although one had been specified? Is the development team responsible for lack of this facility? If not, who is? Why?
4. What backup would you have specified for this system?

maintenance work and so do not personally correct program bugs or latent design errors that surface after the new system is in operation. Convoluted systems logic, software that can only be understood by the programmer who wrote it, missing or incomplete documentation is not their concern. Maintenance is the responsibility of others, usually junior programmers or programming trainees. If the system later requires redevelopment to meet changed environmental conditions, a new team will be formed to do the work. In all likelihood the original analysts and programmers will be assigned to other projects—perhaps they are even working in another company's employ. Though responsible for a design that is difficult to modify or enhance, they will not personally be burdened with the consequences of that design.

The fallacy of separating development from systems maintenance is becoming apparent to all organizations that rely on information processing. Maintenance is costly. It is estimated that 50 per cent of data processing budgets are allocated to maintenance and that maintenance work constitutes more than half of the workload of the typical systems department. (*Systems department* is the name usually given to the unit responsible for systems development and maintenance. The work in this department is discussed in Chapter 18.) More than $30 billion is spent on maintenance annually worldwide. Most organizations report that they spend more for maintenance per instruction than for development. (*See* Fig. 17.2 for systems development and maintenance effort distribution.) No wonder corporate managers are demanding that more attention be paid to ease off maintenance and modification when systems are being designed.



Source: M.V. Zelkowitz, 'Perspectives on Software Engineering', *ACM Computing Surveys* New York: Association for Computing Machinery 10, no. 2, (June 1978), p. 202.

Fig. 17.2 Effort Distribution

This chapter describes how information systems are evaluated once they are operational. Causes of unsatisfactory output are discussed that lead to systems maintenance or redevelopment. The chapter closes with reasons why maintenance is unpopular, ways to improve maintenance, and suggestions for systems design in order to facilitate maintenance.

EVALUATION OF PROCESSING

The *performance* of an information system that is under the jurisdiction of a systems department should be continually monitored by a production manager, who supervises and controls processing activities from scheduling through output distribution. Data may be collected on the relationship of input to output (efficiency of process) and on factors such as availability, flexibility, reliability, timeliness, integrity, accuracy, repairability, and quality of operations (effectiveness of product). Questions will be raised regarding the timeliness of reports, their completeness, and their validity. Do errors and discrepancies exist? Is the system achieving long-range goals? How often is the system down? Does output meet user expectations?

Many performance factors can be measured by monitors that collect quantitative data. For example, hardware monitors can gather data on the utilization of the CPU and peripherals. They can identify bottlenecks, downtime, and saturation conditions of channels and storage devices. Software monitors, such as auditing programs, collect information on how system resources have been used, by whom, and for what applications. In addition, monitor programs embedded in software may keep track of the demand for application programs, software features, and software routines. Such data is useful for evaluation purposes but has a price tag since it must be processed. Monitors can also interfere with optimal operating efficiency.

Another approach to data collection is to solicit the views of people who interact with the information system through surveys, questionnaires, and interviews. Observation and study of documentation, such as logs that record complaints or error frequency, are other ways to assess system performance.

In many organizations, information systems that are the responsibility of the systems department are periodically evaluated: for example, every six months. Systems evaluation may also take place when problems arise or when end-users express dissatisfaction with output or service. In order to fight 'brush fires', unscheduled evaluations are common.

During systems evaluation, performance is compared to the specification of systems requirements and checked to see that departmental standards for information processing are being maintained. For systems judged satisfactory, operations continue as before. Should systems evaluation identify problems that need correction, a maintenance or redevelopment cycle is initiated. In general, *maintenance* is defined as a change that affects few users and one that does not require much effort or many resources: for example, not more than two weeks of a programmer's time. *Redevelopment*, on the other hand, requires a major allocation of resources and personnel.

Figure 17.3 illustrates that both maintenance and redevelopment have a *life cycle* similar to that of systems development. Once the need for maintenance (or redevelopment) has been identified, the cost and effort required to upgrade the system is assessed in a feasibility study. If management approves the project, the work is scheduled and assigned to systems personnel. The existing system is first analyzed using the tools and techniques described in this textbook. Then changes

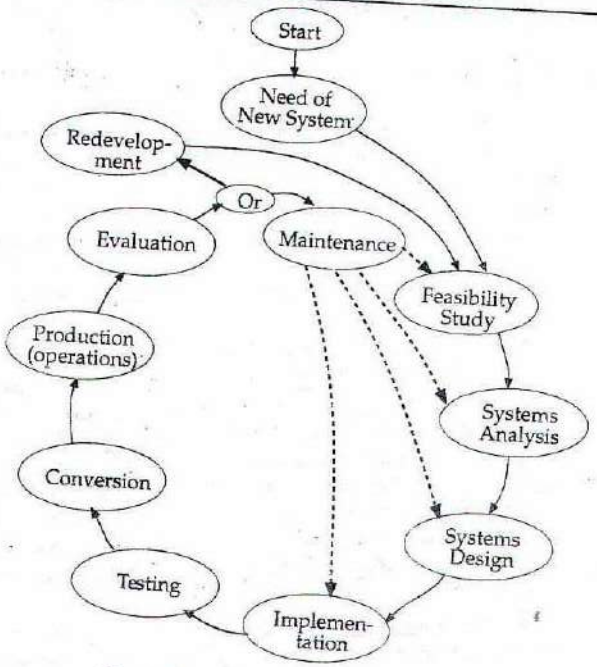


Fig. 17.3 Development Cycle

to the system are designed, implemented, and tested. If test results are satisfactory, conversion to the modified (or redeveloped) system takes place.

Even routine maintenance should not skip steps in the development cycle. To do so can lead to monumental blunders like the mistake that occurred when a programmer modified a grade report program at a university to make a change in output, and is doing so changed statement numbers. Figure 17.4 shows the original statement numbers to the left of each box and modified numbers to the right. Due to the modification being trivial, the programmer implemented the program without testing. Look what happened. Since the 'Go to 10' statement was unaltered but the statement numbers had been changed, the program skipped 'Read grade data on next student'. The result was that Mr L.C. Able, the first name in an alphabetical listing of students, received 14,000 grade reports. Mistakes of this nature can be expensive, disruptive, and ruin an organization's credibility. (This error illustrates not only poor maintenance process, but poor control of output as well, since the error was not caught in mailing.)

WHY DO SYSTEMS NEED MAINTENANCE

Not all jobs are run successfully. Sometimes peripherals jam. An electrical storm

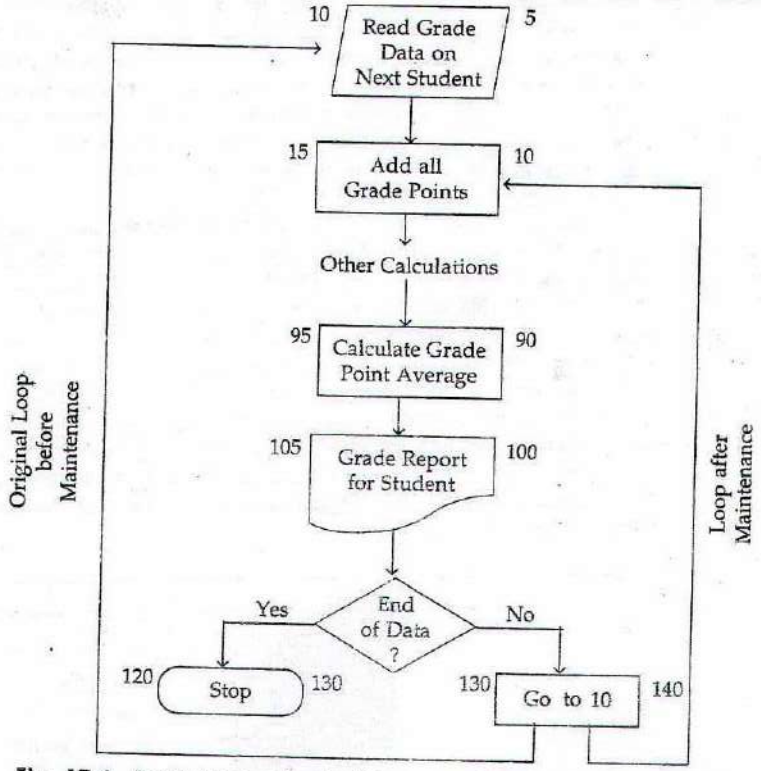


Fig. 17.4 Partial Flowchart of Program Carelessly Maintained

may cause a power failure, which makes the system go down. An airconditioning system may malfunction causing the CPU to overheat and break down. Sometimes an overload or unexpected boundary condition causes an error message to appear on the operator's terminal, sometimes output fails to pass output controls; Sometimes program bugs appear. Whatever the source of the problem, a previously working system that ceases to function requires *emergency maintenance*. Isolating operational problems is not always an easy task, particularly when a combination of circumstances are responsible. The ease with which the problem can be corrected is directly related to how well the system has been designed and documented.

Changes in the environment in which an information system operates may also lead to systems maintenance. For example, new laws or government regulation may require a company to process new reports. Competitors may alter market conditions so that systems in use are no longer appropriate. A new manager may be hired who has a different style of decision making than predecessors so that a different threshold of information is required. Perhaps system users have increased their awareness of the potential of computer technology so now have higher

expectations regarding information delivery. Policies of the organization may change, requiring new methods for calculations, such as new depreciation methods. Information systems should be able to accommodate changing needs. They should be flexible in design so that new features can be added with ease.

Systems personnel may themselves trigger systems maintenance as they detect errors in a system resulting from poor design. Perhaps new hardware has been acquired by the organization that makes it necessary to modify programs or alter protocols. Codes may require revision because of organizational changes. Analysts may have a wish list of features they would like to add to the system to upgrade its performance. Many ideas for systems enhancement are conceived and documented during development but not included in the design because of lack of resources or because the design is frozen. Once the system is operational, the suggestions are renewed.

SETTING MAINTENANCE PRIORITIES

Emergency maintenance and minor system changes are generally authorized by systems department management. But a decision whether to patch an old system or completely redevelop it may be referred to a committee that is made up of representatives from the systems department, end-users, and management. Or perhaps the MIS Planning Committee makes this decision.

Redevelopment may be chosen over maintenance because the system in question has been in use for a long time and may be written in languages that today's programmers are not familiar with, such as assembly language or PL/1. The software may use outdated programming and design practices, factors that complicate maintenance. Programs may have been extensively patched with changes poorly documented, or they may have been increased in size beyond what their structure was originally designed to bear. An old system might be compared to an old car that becomes increasingly unreliable and costly to maintain with age. Besides, most end-users, like car owners, want the latest technology. They favour the use of software that takes advantage of streamlined facilities, standard routines, and enhanced computational and retrieval capabilities.

Since there are typically far more requests for maintenance and redevelopment than a systems department can possibly service, decision on which systems to maintain (or redevelop) are generally based on priority criteria. Ideally, maintenance priorities should be decided on the basis of benefit-cost and or performance-cost ratios. Some firms base maintenance priority on the worst-first rule, clearly a subjective judgment. Too often the assignment of maintenance priority is based on corporate politics, not on economic or technological grounds.

MAINTENANCE MANAGEMENT

In a microcomputer environment, equipment repair may be done by an internal service department. However, many organizations sign a manufacturer's maintenance agreement or a contract with a third-party maintenance firm for hardware upkeep.

With regard to software maintenance, the vendor of a package may be called for help. End-users are responsible for the maintenance of programs that they write themselves. Information centres staffed by data processing professionals have been established in many organizations to help end-users manage microcomputer resources. (*See* Chapter 20.) Indirectly, information centres do provide maintenance assistance. For example, a course may be given on how to program using the language Pascal in which the instructor explains how to modify instructions in order to upgrade programs. Through exercises in rewriting code, end-users gain maintenance experience and, in the process, learn ways to write programs that can be easily maintained.

Systems departments have personnel assigned to maintenance of computer systems in their jurisdiction. Minor hardware repair may be done by internal staff, but for a major breakdown, an outside maintenance firm will probably be called. Most organizations sign a manufacturer's maintenance agreement, for which they pay a fixed monthly fee, or contract maintenance service with a third-party maintenance firm. The majority of the people assigned to maintenance within the department are analysts and programmers who work to keep software running.

Although software maintenance takes a large share of the total data processing budget, it is surprising how few companies make a serious effort to reduce software maintenance costs. Typically maintenance is a low status job delegated to junior level personnel whose inexperience may prolong the length of time it takes to trace and resolve problems, which adds to maintenance costs. Surveys reveal that maintenance is regarded as non-creative and non-challenging with only half the motivating potential of other analysis/programming work. Experienced analysts and programmers prefer systems development and use their seniority to get such assignments.

Unfortunately management contributes to this state of affairs. People assigned to maintenance seldom receive professional recognition for their contributions. One indication of this fact is that few companies have even established the job classification of maintenance analyst or maintenance manager. The record shows that job dissatisfaction and turnover is high in systems maintenance, higher than in other classifications of technical data processing personnel.

Clearly, improved maintenance productivity is a way to lower maintenance costs. This might be done by:

- Hiring people with a flair for detective work who prefer systems maintenance over other types of work.
- Enhancing maintenance jobs and giving them status and remuneration that attracts qualified and experienced personnel.

The first suggestion is often impractical because of lack of candidates. The second should be the focus of management. There are five job dimensions that typically motivate data processing professionals—skill variety, task identification, task significance, autonomy, and feedback from the job. If systems maintenance is enhanced in these areas, it is likely that the job will become attractive to senior

analysts and programmers. For example, maintenance jobs might be rotated, giving analysts and programmers experience with different types of systems. In this way, the job might become more challenging while providing valuable backup to the organization. Maintenance work should be well paid and, in addition, quality work should be recognized and rewarded, perhaps with a monetary bonus. A career track should be established to allow for professional advancement of maintenance personnel. Maintenance personnel should be given a role in decision making, perhaps assigned to the committee that sets maintenance priorities or assigned to development teams to ensure that ease of maintenance is incorporated in systems design.

MAINTENANCE GUIDELINES

As you have learned in this chapter, systems maintenance and redevelopment are to be expected. Equipment malfunctions and breaks down from time to time. Latent software design errors or program bugs may surface and require correction. Systems have to be updated in response to changed conditions or upgraded when changes in hardware, software, or protocols take place. User specifications themselves may change, requiring corresponding changes in programs and systems design. Knowing this, a team responsible for the development of a new information system should design the system and implement it in ways that facilitate maintenance. In addition, the work of the team should be of high quality so that once a system is operational there is no call for maintenance that might have been avoided.

What are the characteristics of information systems that are easy to maintain?

- *The systems are planned with an eye to the future*

Members of the MIS Planning Committee and analysts assigned to the development team look ahead, anticipate growth of the organization's information needs and advances in technology. A system designed for tomorrow will not become outdated as quickly as one that focuses on present needs alone.

- *User specifications are correct*

The development team does not rush through the front end of the development cycle in order to produce evidence of progress in the form of program code. Instead, time is taken to review the problem, to listen to users, and to study the environment in which the new system will operate in order to ensure that specifications are accurate and complete. This avoids a too common maintenance problem: the need to modify an information system immediately following conversion in order to incorporate omitted features that prove indispensable.

- *The system is modular*

It is much easier to test components of a system when searching for the source of errors when a system has a modular design. In addition, maintenance personnel can work in parallel to enhance the system or repair defects. A modular structure also lends itself to expansion. Most information systems need to grow over time in order to keep pace with changes in the organizational environment they support.

- *Documentation is complete*

In order to trace a problem or upgrade a system, maintenance personnel must understand the system. They turn to systems documentation to learn about its design and to acquire the information they need to make system modifications. Imagine the problems that arise when all knowledge about a system is centred in a few individuals, not recorded in a documentation package. These individuals may no longer remember details of the development project or may no longer be in the company's employ? Maintenance costs can skyrocket without adequate documentation because of the time it takes for maintenance personnel to figure out how a system works.

- *Standards are followed during development*

Standards contribute to productivity by providing guidelines for the way in which work should be done. They also promote communication among members of the development team and end-users, and ensure compatibility of system modules. In all likelihood, a development team that follows standards will produce a more efficient and effective system than one that does not. As a result, the system will need less maintenance. In addition, standards help those who will subsequently operate and maintain the system understand how the system functions.

- *Testing is thorough*

Poorly designed test cases, incomplete test data, haste in testing so that test results are not carefully evaluated, or failure to retest when changes are made to the system's design can all have the result that errors pass undetected only to surface once conversion takes place. Since comprehensive testing may not always be practical, the aim should be careful test design so that most errors will be caught.

- *Adequate time is allowed for the development cycle*

When time is constrained and both corporate management and end-users put excessive pressure on the development team to produce results, the development team may be forced to produce a system in haste. Perhaps integration with other systems has been abandoned. Perhaps not all users specifications have been met. Perhaps testing is short-changed. Perhaps quality suffers because time was not taken to plan the best possible design. Whenever a development team sacrifices quality because of time pressures, there is a corresponding increase in the amount of maintenance that the system requires at a later date.

- *Attention is paid to end-users and consideration of ergonomics and human factors*

The purpose of an information system is to assist humans in their work. If it is difficult or uncomfortable for people to interact with a computer system, there will be complaints and requests for system modification. The novelty of computer technology has worn off. Most users today are much less tolerant of design peccadillos than in the past and also have higher expectations regarding computing services. There will be fewer requests for maintenance if user-friendly features are incorporated in systems design.

- *Awareness exists on the part of the development team regarding the relationship of systems design and systems maintenance*

Most important of all, members of the development team should continually ask themselve—how can this idea, this feature, this program code be economically

maintained? With the number of information systems on the rise and the complexity of these systems increasing, the issue of maintenance is more pressing than in the past. Brilliant, complex code that no one, other than the original programmer can understand, much less maintain, has no place in modern information systems. The attitude that documentation is a waste of time and beneath the dignity of analysts and programmers must be changed. Ease of systems maintenance should be a primary design objective for every information system. Of course, if members of development team are well qualified, experienced, and dedicated to excellence, corrective maintenance can be minimized. Selection of the development team is therefore a crucial management decision. The organization of the development team, project management, and end-user involvement also have a bearing on the quality of an information system. Part Five in this book is a discussion of these issues.

SUMMING UP

Systems maintenance is very costly and takes an effort much greater than the effort required for systems development. It is not surprising that organizations that rely on information processing are searching for ways to reduce this effort. An obvious solution is to improve the design of information systems so that errors of logic or mistakes in programming do not occur. Another solution is to plan systems to facilitate maintenance. The development of systems that are modular is one way to facilitate maintenance. Still another solution to the problem of maintenance is to anticipate information needs when a system is under development. This will mean that systems do not quickly become outdated.

Figure 17.5 shows the relationship of maintenance to systems evaluation. Note that

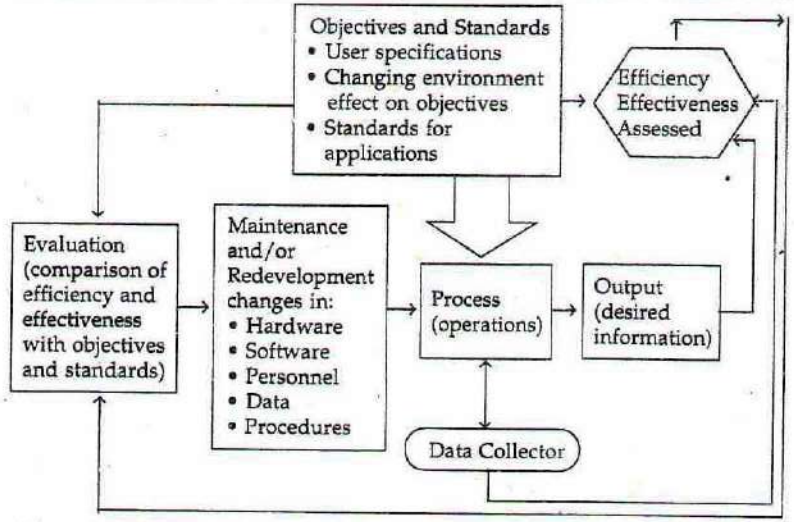


Fig. 17.5 Efficiency and Effectiveness of Computer Application

effectiveness (the ability of the system to meet operational demands within time constraints) is measured by comparing output with system specifications and standards. In addition, the current operating environment is assessed in which the output is used. Efficiency (the ratio of useful work performed to the total energy expended) is measured by monitors and by the collection of data from people who interact with the system. Unsatisfactory performance may trigger maintenance or redevelopment in hardware, software, personnel, procedures, or data. Then processing takes place followed by a new cycle of evaluation and maintenance.

It is hoped you have learned about problems of maintenance in reading this chapter and have a better understanding of maintenance concerns. This understanding is important for end-users and management as well as for systems personnel. Each of these groups plays a role in systems development. Each bears a responsibility for quality design and implementation so that corrective maintenance is minimal. Each should monitor the development process to make sure that ease of maintenance is built into systems design.

CASE 39

SURVEY FINDS MAINTENANCE PROBLEMS SEVERE

According to survey results released by the Quality Assurance Institute in 1986, the software maintenance problem within large data centres has yet to be solved. Here are some of the highlights of the survey, which polled 37 Fortune 500-class companies.

- Maintenance backlogs in the companies surveyed ranged from two months to 60 months—the average was 23 months.
- Expenditures of maintenance ranged from 10 per cent to 90 per cent of data processing budgets—the average was 51 per cent.
- Nearly 80 per cent of the respondents had systems whose logic could only be understood by specific individuals. This prevented rotating maintenance responsibility among data processing personnel.
- Formal methods for deciding when to rewrite programs existed in less than 15 per cent of the companies surveyed. The requirement that older systems conform to the same programming standards as newly developed systems was found in 16 per cent of the firms.
- In all but 5 per cent of the firms surveyed, it was acknowledged that a programmer working on new systems development had a more prestigious position than one assigned to maintenance.

Source: John Gallant, 'Survey Finds Maintenance Problem Still Escalating,' *Computerworld*, Vol. 20, no. 27, (Jan. 27, 1986), p.31.

Questions

1. The survey reveals that programmers working on new systems have a more prestigious position than programmers working on maintenance? Should this difference exist? Does it benefit the firm?
2. Are you surprised at the results of the survey? Appalled? Explain.