CHAPTER 10 Simple Implementation And Maintanance

LEARNING OUTCOMES

By the end of this chapter, you should be able to:

1. Explain different types of testing including unit, integration, system and acceptance;
2. Apply four installation strategies;
3. Differentiate between program, system, operation and user documentation;
4. Compare modes available for system training including supplier, external or in-house;
5. Explain and contrast four types of maintenance; and
6. Describe maintenance management issues, including personnel, quality measurement and controlling change requests.

INTRODUCTION

System implementation and maintenance are the last two phases of the systems development life cycle. The purpose of implementation is to build a properly working system, test and install it in the organisation, finalise documentation and train the users. In this chapter you will learn various type of testing to ensure the quality of the system developed. Once the system accepted by the user, the system will be installed with several approaches installation strategies. You will also learn the different types of documentation and training for the user to use the system effectively.

The purpose of maintenance is to fix and enhance the system to respond to problems and changing business conditions. Maintenance includes activities from all systems development phase. Maintenance also involves responding to requests to change the system, transforming request into changes, designing the changes and implementing them. This chapter describes four kinds of maintenance types: corrective, adaptive, perfective and preventive. Finally we describe the process in managing maintenance which involves maintenance personnel, measuring maintenance effectiveness and controlling maintenance requests.

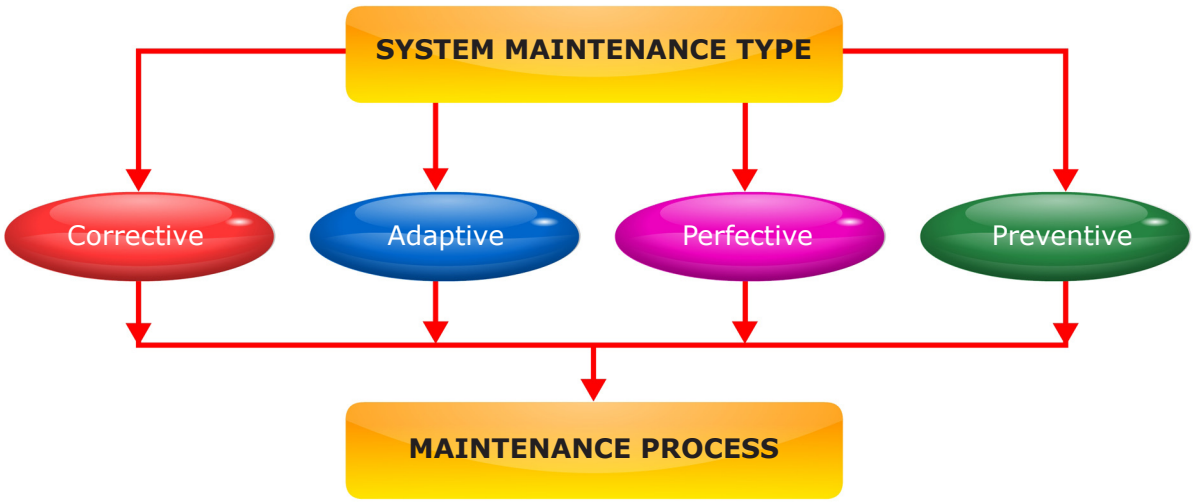


Figure 10.1: Examples of sources for finding information

10.1 SYSTEM IMPLEMENTATION

System implementation is the delivery of the system into production (day-to-day operation). It is made up of many activities. There are several tasks involved in the implementation phase for a typical systems development project which are coding, testing, installation, documentation and training.

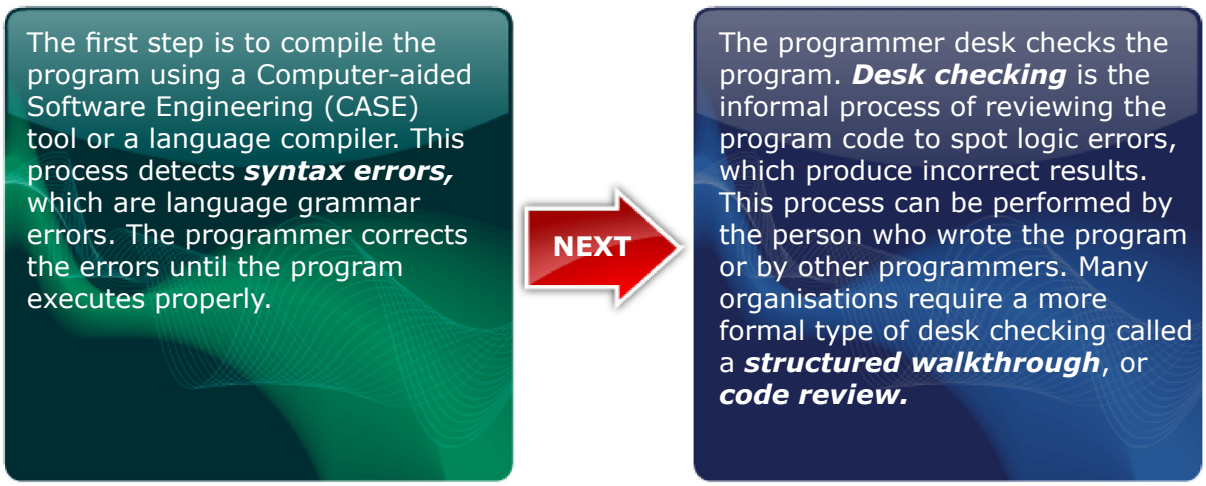
Coding, as we mentioned before, is the process whereby the physical design specifications created by the analysis team are turned into working computer code by the programming team. Depending on the size and complexity of the system, coding can be an involved, intensive activity. Regardless of the development methodology followed, once coding has begun, the testing process can begin and proceed in parallel. As each program module is produced, it can be tested individually, then as part of a larger program, and then as part of a larger system. You will learn about the different strategies for testing later in this chapter.

Installation is the process during which the current system is replaced by the new system. This includes conversion of existing data, software, documentation and work procedures to those consistent with the new system. Although the process of documentation proceeds throughout the life cycle, it receives formal attention during the implementation phase because the end of implementation largely marks the end of the analysis team's involvement in systems development. As the team is getting ready to move on to new projects, the team need to prepare documents that reveal all of the important information accumulated during its development and implementation.

The result of installation and documentation process is training to the user so that they can quickly learn the new system. Larger organisations tend to provide training to computer users which are very specific to particular application system, whereas the rest is general to particular operating systems or off-the-shelf software packages. Each of tasks in system implementation phase is addressed in more detail later in this chapter.

10.2 TESTING

After coding, a programmer must test each program to make sure it functions correctly. Later, programs are tested in groups, and finally the development team must test the entire system.



Typically, a group of three to five IT staff members participate in code review. The group usually consists of project team members and might include other programmers and analysts who did not work on the project. The objective is to have a peer group identify errors, apply quality standards, and verify that the program meets the requirements of the system design specification. Errors found during a structured walkthrough are easier to fix while coding is still in the developmental stages.

In addition to analysing logic and program code, the project team usually holds a session with users called a design walkthrough, to review the interface with a cross section of people who will work with the new system and assure that all necessary features have been included. This is a continuation of the modeling and prototyping effort that began early in the systems development process. The next step in application development is to initiate a sequence of unit testing, integration testing, system testing and acceptance testing.

10.2.1 Unit Testing

The testing of an individual program or module is called unit testing or known as module testing. The objective is to identify and eliminate execution errors that could cause the program to terminate abnormally, and logic errors that could have been missed during desk checking.

Test data should contain both correct data and erroneous data and should test all possible situations that could occur.

For example;

For a field that allows a range of numeric values, the test data should contain minimum values, maximum values, values outside the acceptable range, and alphanumeric characters. During testing, programmers can use software tools to determine the location and potential causes of program errors.

During unit testing, programmers must test programs that interact with other programs and files individually, before they are integrated into the system. This requires a technique called stub testing. In stub testing, the programmer simulates each program outcome or result and displays a message to indicate whether or not the program executed successfully. Each stub represents an entry or exit point that will be linked later to another program or data file.

10.2.2 Integration Testing

Testing two or more programs that depend on each other is called integration testing, or link testing.

For example;

Consider an information system with a program that checks and validates customer credit status, and a separate program that updates data in the customer master file. The output from the validation program becomes input to the master file update program. Testing the programs independently does not guarantee that the data passed between them is correct. Only by performing integration testing for this pair of programs can you make sure that the programs work together properly.

Systems analysts usually develop the data they use in integration testing. As is the case with all forms of testing, integration test data must consider both normal and unusual situations. For example, integration testing might include passing typical record between two programs, followed by blank records, to simulate an unusual event or an operational problem. You should use test data that simulates actual conditions because you are testing the interface that links the programs. A testing sequence should not move to the integration test stage unless it has performed properly in all unit tests.

10.2.1 Unit Testing

After completing integration testing, you must perform system testing, which involves the entire information system, A system test includes all typical processing situations and is intended to assure users, developers, and managers that the program meets all specifications and that all necessary features have been included. During a system test, users enter data, including samples of actual, or live, data, perform queries, and produce reports to simulate actual operating conditions. All processing options and outputs are verified by users and the IT project development team to ensure that the system functions correctly.

10.2.4 Acceptance Testing

Once the system tests have been satisfactorily completed, the system is ready for acceptance testing which is testing the system in the environment where it will eventually, be used. Acceptance refers to the fact that users typically sign off on the system and "accept" it once they are satisfied with it. As we said previously, the purpose of acceptance testing is for users to determine whether the system meets their requirements. The extent of acceptance testing will vary with the organisation. The most complete acceptance testing will include alpha testing and beta testing. Alpha testing is simulated or actual system testing by potential users or an independent test team at the developers' site. Beta testing comes after alpha testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs.

There are several test performed during alpha testing include the following:

- Recovery testing - forces the software (or environment) to fail in order to verify that recovery is properly performed.
- Security testing - verifies that protection mechanisms built into the system will protect it from improper penetration.
- Stress testing - tries to break the system (e.g., what happens when a record is written to the database with incomplete information or what happens under extreme online transaction loads or with a large number of concurrent users).
- Performance testing - determines how the system performs in the range of possible environments in which it may be used (e.g., different hardware configurations, networks, operating systems, and so on). Often the goal is to have the system perform with similar response time and other performance measures in each environment.

In beta testing, a subset of the intended users runs the system in their own environments using their own data. The intent of the beta test is to determine whether the software, documentation, technical support, and training activities work as intended. Beta testing can be viewed as a rehearsal of the installation phase. Problems uncovered in alpha and beta testing in any of these areas must be corrected before users can accept the system.

10.3 INSTALLATION

When a new system has been completely developed and tested, it needs to be installed at locations it is supposed to run. The process of installation is a complex one as it involves users who need to leave behind the old system and begin to use the new system completely. Installation will confirm whether or not the new system can perform the functions that have been planned for it. Four different approaches to installation as shown in Table 10.1 have emerged over the years: direct, parallel, pilot and phased.

Figure 10.1: Examples of sources for finding information

Approaches	Description
Direct Installation	With direct installation (sometimes called cold turkey or abrupt cutover), the new system instantly replaces the old system. The new system is turned on and the old system is immediately turned off. This is the approach that you probably use when you upgrade commercial software (e.g., Microsoft Word) from one version to another; you simply begin using the new version and stop using the old version. Direct installation is the simplest and most straightforward. However, it is the most risky, because any problems with the new system that have escaped detection during testing may seriously disrupt the organisation. On the other hand, it is the least expensive installation method, and it creates considerable interest in making the installation a success.
Parallel Installation	With parallel installation, the new system is operated side by side with the old system, both systems are used simultaneously. For example, if a new accounting system is installed, the organisation enters data into both the old system and the new system and then carefully compares the output from both systems to ensure that the new system is performing correctly. After some time period (often one to two months) of parallel operation and intense comparison between the two systems, the old system is turned off and the organisation continues using the new system. This approach is more likely to catch any major bugs in the new system and prevent the organisation from suffering major problems.

	If problems are discovered in the new system, the system is simply turned off and fixed and then the installation process starts again. The problem with this approach is the added expense of operating two systems that perform the same function.
Pilot Installation	With a pilot installation, one or more locations in units or work groups within a location are selected to be installed first as part of a pilot test. The locations participating in the pilot test are installed (using either direct or parallel installation). If the system passes the pilot test, then the system is installed at the remaining locations (again using either direct or parallel installation). Pilot installation has the advantage of providing an additional level of testing before the system is widely deployed throughout the organisation, so that any problems with the system affect only the pilot locations. However, this type of installation obviously requires more time before the system is installed at all organisational locations. Also, it means that different organisational units are using different versions of the system and business processes, which may make it difficult for them to exchange data.
Phased Installation	With phased installation, the system is installed sequentially at different locations, A first set of locations are installed, then a second set, then a third set, and so on, until all locations are installed. Sometimes there is a deliberate delay between the different sets (at least between the first and the second), so that any problems with the system are detected before too much of the organisation is affected. In other cases, the sets are installed back-to-back so that as soon as those installing one location have finished, the project team moves to the next and continues the installation. Phased installation has the same advantages and disadvantages of pilot installation. In addition, it means that a smaller set of people are required to perform the actual installation than if all locations were installed at once.

10.3.5 Selecting the Appropriate Installation Strategy

There are three important factors to consider in selecting an installation strategy: risk, cost, and the time required

Table 10.2: Characteristics of InstallationStrategies

Characteristic	Direct	Parallel	Pilot	Phased
Risk	High	Low	Low	Medium
Cost	Low	High	Medium	Medium
Time	Short	Long	Medium	Long

(a) **Risk**

Even after all the tests, there may still be a few undiscovered bugs or errors. The installation process provides one last step in which to catch these bugs before the system goes live and the bugs have the chance to cause problems.

Parallel installation is less risky than is direct installation because it has a greater chance of detecting bugs that have gone undiscovered in testing. Likewise, pilot installation is less risky than is phased installation because if bugs do occur, they occur in pilot test locations whose staffs are aware that they may encounter bugs. Since potential bugs affect fewer users, there is less risk. Likewise, installing a few modules at a time lowers the probability of a bug because there is more likely to be a bug in the whole system than in any given module.

How important the risk is depends on the system being implemented, the combination of the probability that bugs remain undetected in the system and the potential cost of those undetected bugs. However, there still might have been mistakes made in the analysis process, so that although there might be no software bugs, the software might fail to properly address the business needs.

(b) **Cost**

As might be expected, different installation strategies have different costs. These costs can include things such as salaries for people who work with the system (e.g., users, trainers, system administrators, and consultants), travel expenses, operation expenses, communication costs, and hardware leases. Parallel installation is more expensive than direct cutover because it requires that two systems (the old and the new) be operated at the same time. Employees must now perform twice the usual work because they have to enter the same data into both the old and the new systems. Parallel installation also requires the results of the two systems to be completely cross-checked to make sure there are no differences between the two, which entails additional time and cost. Pilot installation and phased installation have somewhat similar costs.

(c) Time

The final factor is the amount of time required to install between the old and the new system. Direct installation is the fastest because it is immediate. Parallel installation takes longer because the full advantages of the new system do not become available until the old system is turned off. Phased installation usually takes longer than pilot installation because usually (but not always) once the pilot test is complete all remaining locations are simultaneously installed. Phased installation proceeds in waves, often requiring several months before all locations are installed.

SELF-CHECK

What are the four approaches to installation? Which is the most expensive? Which is the most risky?

10.4 DOCUMENTATION

Documentation is an activity of recording all the specifications and facts about an information system as a reference for the future.

Accurate documentation can reduce system downtime, cut costs and speed up maintenance tasks. A system that has completely gone into operation will not remain permanent just like that. From time to time, there will to be changes being made. The programmer who does the change on the program requires accurate documentation. Besides that, the operators and users working on the system throughout the system's life also need to refer to the documentation as shown in Figure 10.1.

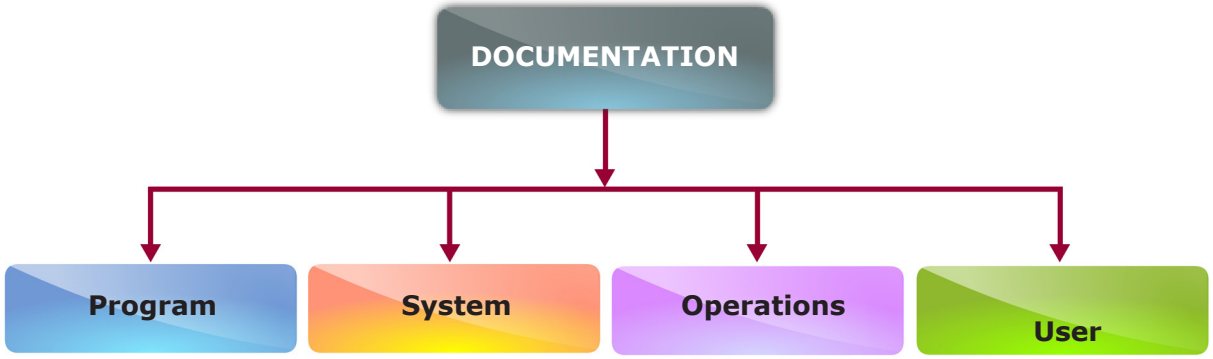


Figure 10.1: Type of documentation

10.4.1 Program Documentation

Program documentation describes the inputs, outputs, and processing logic for all program modules. The program documentation process starts in the systems analysis phase and continues during systems implementation. Systems analysts prepare overall documentation, such as process descriptions and report layouts, early in the SDLC. This documentation guides programmers, who construct modules that arc well supported by internal and external comments and descriptions that can be understood and maintained easily. A systems analyst usually verifies that program documentation is complete and accurate.

10.4.2 System Documentation

System documentation describes the system's functions and how they are implemented. System documentation includes data dictionary entries, data flow diagrams, object models, screen layouts, source documents, and the systems request that initiated the project. System documentation is necessary reference material for the programmers and analysts who must support and maintain the system.

Most of the system documentation is prepared during the systems analysis and systems design phases. During the systems implementation phase, an analyst must review prior documentation to verify that it is complete, accurate, and up-to-date, including any changes made during the implementation process.

For example;

If a screen or report has been modified, the analyst must update the documentation. Updates to the system documentation should be made in a timely manner to prevent oversights.

10.4.3 Operations Documentation

Operations Documentation contains all the information needed for processing and distributing online and printed output. Typical operations documentation includes:

- Scheduling information for printed output eg. Report run frequency and deadlines.
- Input files and where they originate and output files and destinations.
- E-mail and report distribution lists.
- Special forms required, including online forms.
- Error and informational messages to operators and restart procedures.
- Special instructions, such as security requirements.

Operations documentation should be clear, concise, and available online if possible. If the IT department has an operations group, you should review the documentation with them, early and often, to identify any problems. If you keep the operations group informed at every phase of the SDLC, you can develop operations documentation as you go along.

10.4.4 User Documentation

User documentation consists of instructions and information to users who will interact with the system such as user manuals, help screens, Frequently Asked Questions (FAQs) and tutorials. User documentation needs to be written in a language that is easily understood by users. The use of special terms that are too technical need to be avoided because users do not know anything about system development.

An excerpt of online user documentation for Microsoft Word appears in Figure 10.2. Notice how the documentation is organised by topic, each of which has several subtopics underneath. This particular page is devoted to provide assistance and the menu at the left provides links to other relevant pages like training and templates. Such presentation methods have become standard for help files in online personal computer documentation.

Figure 10.2: Example of online user documentation
Source: www.microsoft.com

What is the difference between system documentation and user documentation?

10.5 TRAINING

No system can be successful without proper training. A successful information system requires training for users, managers and staff members. The entire system development effort can depend on whether or not people understand the system and know how to use it effectively.

The type of training needed will vary by system type and user expertise. The lists of potential topics from which you will determine if training will be useful include the following:

- Use of the system (e.g., how, to enter a class registration request).
- General computer concepts (e.g., computer files and how to copy them).
- Information system concepts (e.g., batch processing).
- Organisational concepts (e.g., inventory accounting).
- System management (e.g., how to request changes to a system).
- System installation (e.g., how to reconcile current and new systems during phased installation).

As you can see from this partial list, many potential topics go beyond simply how to use the new system. It may be necessary for you to develop trainings for users in other areas so that users will be ready, conceptually and psychologically, to use the new system. Some training, such as concept training, should begin early in the project because this training can assist in the "unfreezing" (helping users let go of long established work procedures) element of the organisational change process. Each element of training can be delivered in a variety of ways. The main choices are to obtain training from supplier or vendors, external training resources and in house training.

10.5.1 Supplier Training

Most of the software and hardware suppliers offer training programmes for free, or for a very suitable price together with the goods they sell. Companies can also negotiate on the training cost with the supplier by considering the relationship with suppliers and the long-term prospect of a relationship.

Training that is done at a supplier's place is normally conducted by an expert in the

field, with long practical experience. However, if the training involves many participants from one organisation, it can be negotiated to be conducted inside the organisation itself. Supplier training focuses on the product that is developed by the supplier itself. Therefore, the scope of training is limited to the software or hardware version that is owned by the supplier.

10.5.2 External Training Resources

If supplier training is not practical and your organisation does not have the internal resources to perform the training, you might find that outside training consultants are a desirable alternative. Many training consultants, institutes and firms are available that provide either standardised or customised training packages.

10.5.3 In-House Training

One effective strategy for training on a new system is to first train a few key users and then organise training program in house that involve these users to provide further training, both formal and on demand. Often, training is most effective if you customise it to particular user groups and the lead trainers from these groups are in the best position to provide this training to their colleagues.

When developing a training program, several guidelines need to be followed.

Conduct the training in groups with different training programmes for different groups. Group training saves time and training resources. Training sessions need to take into account the background, skills, and type of work of the participants from different groups.

Choose the site that is most effective to conduct the training. Conducting training in the company itself has several advantages such as reduction of travelling cost, and training is done in the real environment in which the system operates. Disadvantages of this method include training participants may be disturbed by things related to real jobs. The use of computer resources for training may disturb company's operation and this will reduce the opportunity for practical training.

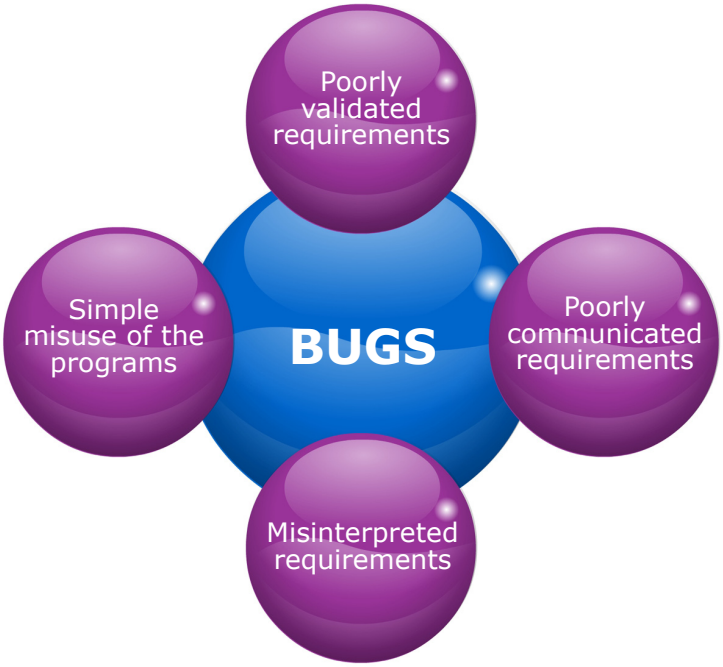
Ensure that listening, seeing, and practical elements are included in the training session. The way a person learns something is different from one individual another. As far as is possible, a training programme needs to fulfill every different method of learning.

Get ready with effective training materials and interactive tutorials. Training material that is user friendly will produce effective training and will become a useful resource for users. Training material can be in the form of training manuals, handouts, or online material.

10.6 SYSTEM MAINTENANCE

System maintenance includes any change, fixing or enhancing to the system after it has been completely developed and installed. Maintenance involves changes in various aspects. Making changes to a system in operation can be more complex than to make changes during system development. This is because any change on the system in operation would affect users, customers and the organisation. A significant portion of the expenditure for information systems within organisations does not go to the development of new systems but to the maintenance of existing system.

Regardless of how well designed, developed and implemented a system or application may be errors, defect or bugs will inevitably occur. Bugs can be caused by any of the following:



The fundamental objectives of system maintenance are:

- To make predictable changes to existing system to correct errors that were made during the systems design or implementation.
- To preserve those aspects of the programs that were correct and to avoid the possibility that "fixes" to programs cause other aspects of those programs to behave differently.
- To avoid as much as possible, degradation of system performance. Poor system maintenance can gradually erode system throughput and response time.

- To complete the task as quickly as possible without sacrificing quality and reliability. Few operational information systems can afford to be down for any extended period. Even a few hours can cost millions of dollars.

To achieve these objectives, you need an appropriate understanding of the programs you are fixing and in which those programs participate. Lack of this understanding is often the downfall of systems maintenance. We will discuss later various types of maintenance and how to manage the maintenance activities. Given that system maintenance consume the majority of system expenditures, gaining an understanding of these topics will yield numerous benefits to your career as an information systems professional.

10.6 TYPES OF MAINTENANCE

There are several types of system maintenance such as corrective, adaptive, perfective and preventive. Explanation on the four types of maintenance shows in Table 10.3.

Table 10.3 Types of Maintenance

Type	Description
Corrective Maintenance	**Corrective maintenance** is changes made to a system to repair flaws in its design, coding or implementation. For example, if you had recently purchased a new home, corrective maintenance would involve repairs made to things that had never worked as designed, such as a faulty electrical outlet or a misaligned door. Most corrective maintenance problems surface soon after installation. When corrective maintenance problems surface, they are typically urgent and need to be resolved to curtail possible interruptions in normal business activities. Of all types of maintenance, corrective maintenance takes as much as 75 percent of all maintenance activity. This is unfortunate because corrective maintenance adds little or no value to the organisation, it simply focuses on removing defects from an existing system without adding new functionality.
Adaptive Maintenance	**Adaptive maintenance** is changes made to a system to evolve its functionality to changing business needs or technologies. Within a home, adaptive maintenance might be adding an air conditioner to improve air circulation. Adaptive maintenance is usually less urgent than corrective maintenance because business and technical changes typically occur over some period of time. Contrary to corrective maintenance, adaptive maintenance is generally a small part of an organisation's maintenance effort, but it adds value to the organisation.

Perfective Maintenance	**Perfective maintenance** is changes made to a system to add new features or to improve performance. It involves making enhancements to improve processing performance or interface usability or to add desired, but not necessarily required. In our home example, perfective maintenance would be adding a new room. Many systems professionals feel that perfective maintenance is not really maintenance but rather new development.
Preventive Maintenance	**Preventive maintenance** is changes made to a system to avoid possible future problems. An example of preventive maintenance might be to increasethe number of records that a system can process far beyond what is currently needed or to generalise how a system sends report information to a printer so that the system can easily adapt to changes in printer technology. In our home example, preventive maintenance could be painting the exterior to better protectthe home from severe weather conditions. As with adaptive maintenance, both perfective and preventive maintenance aretypically a much lower priority than corrective maintenance. Over the life of a system, corrective maintenance is most likely to occur after initial system installation or after major system changes. This means that adaptive, perfective, and preventive maintenance activities can lead to corrective maintenance activities if not carefully designed and implemented.

10.8 MANAGING MAINTENANCE

As maintenance activities consume more and more of the systems development budget, maintenance management has become increasingly important. In other words, maintenance is the largest segment of programming personnel and this implies the need for careful management. We will address this concern by discussing several issues related to the effective management of systems maintenance.

10.8.1 Managing Maintenance Personnel

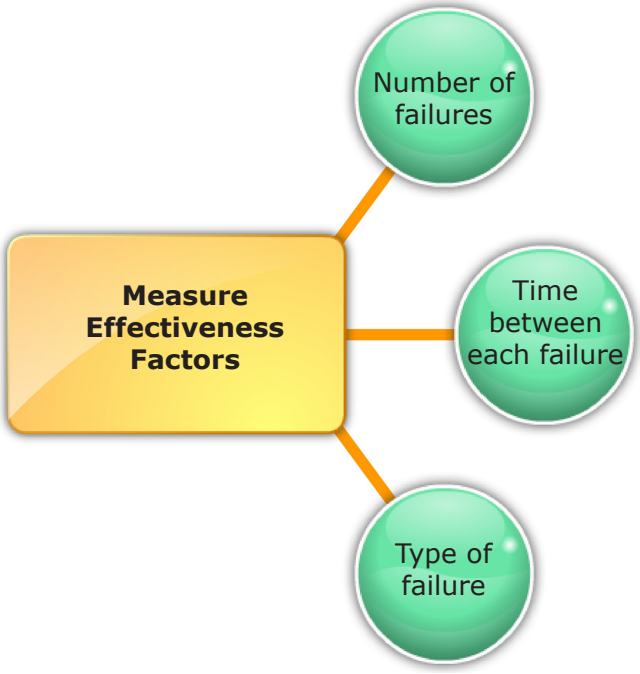
One concern with managing maintenance relates to personnel management. Among the issues arise were should the maintenance group be separated from the development group? Or should the same people who build the system also maintain it? A third option is to let the primary end users of the system in the functional units of the business have their own maintenance personnel. The advantages and disadvantages to each of these organisational structures are summarised in Table 10.4.

Table 10.4 Advantages and Disadvantages of Different Maintenance Organisational Structures

Type	Advantages	Disadvantages
Separate	Formal transfer of system between groups improves the system and documentation quality.	All things cannot be documented, so the maintenance group may not know critical information about the system.
Combined	Maintenance group knows or has access to all assumptions and decisions behind the system's original design.	Documentation and testing thoroughness may suffer due to a lack of a formal transfer of responsibility.
Functional	Personnel have a vested interest in effectively maintaining the system and have a better understanding of functional requirements	Personnel may have limited job mobility and lack access to adequate human and technical resources.

10.8.2 Measuring Maintenance Effectiveness

A second management issue is the measurement of maintenance effectiveness. As with the effective management of personnel, the measurement of maintenance activities is fundamental to understanding the quality of the development and maintenance efforts. To measure effectiveness, you must measure the following factors:



Operations documentation should be clear, concise, and available online if possible. If the IT department has an operations group, you should review the documentation with them, early and often, to identify any problems. If you keep the operations group informed at every phase of the SDLC, you can develop operations documentation as you go along.

For example;

Knowing that a system repeatedly fails to log new account information to the database when a particular customer is using the system can provide invaluable information to the maintenance personnel. Were the users adequately trained? Is there something unique about this user? Is there something unique about an installation that is causing the failure? What activities were being performed when the system failed?

Tracking the types of failures also provides important management information for future projects.

For example;

If a higher frequency of errors occurs when a particular development environment is used, such information can help guide personnel assignments, training courses, or the avoidance of a particular package, language, or environment during future development. The primary lesson here is that without measuring and tracking maintenance activities, you cannot gain the knowledge to improve or know how well you are doing relative to the past. To effectively manage and to continuously improve, you must measure and assess performance over time.

10.8.3 Controlling Maintenance Requests

Another maintenance activity is managing maintenance requests. There are various types of maintenance requests, some correct minor or severe defects in the system whereas others improve or extend system functionality. From a management perspective, a key issue is deciding which requests to perform and which to ignore. Because some requests will be more critical than others, some method of prioritising requests must be determined. Figure 10.3 shows a flowchart that suggests one possible method you could apply for dealing with maintenance change requests.

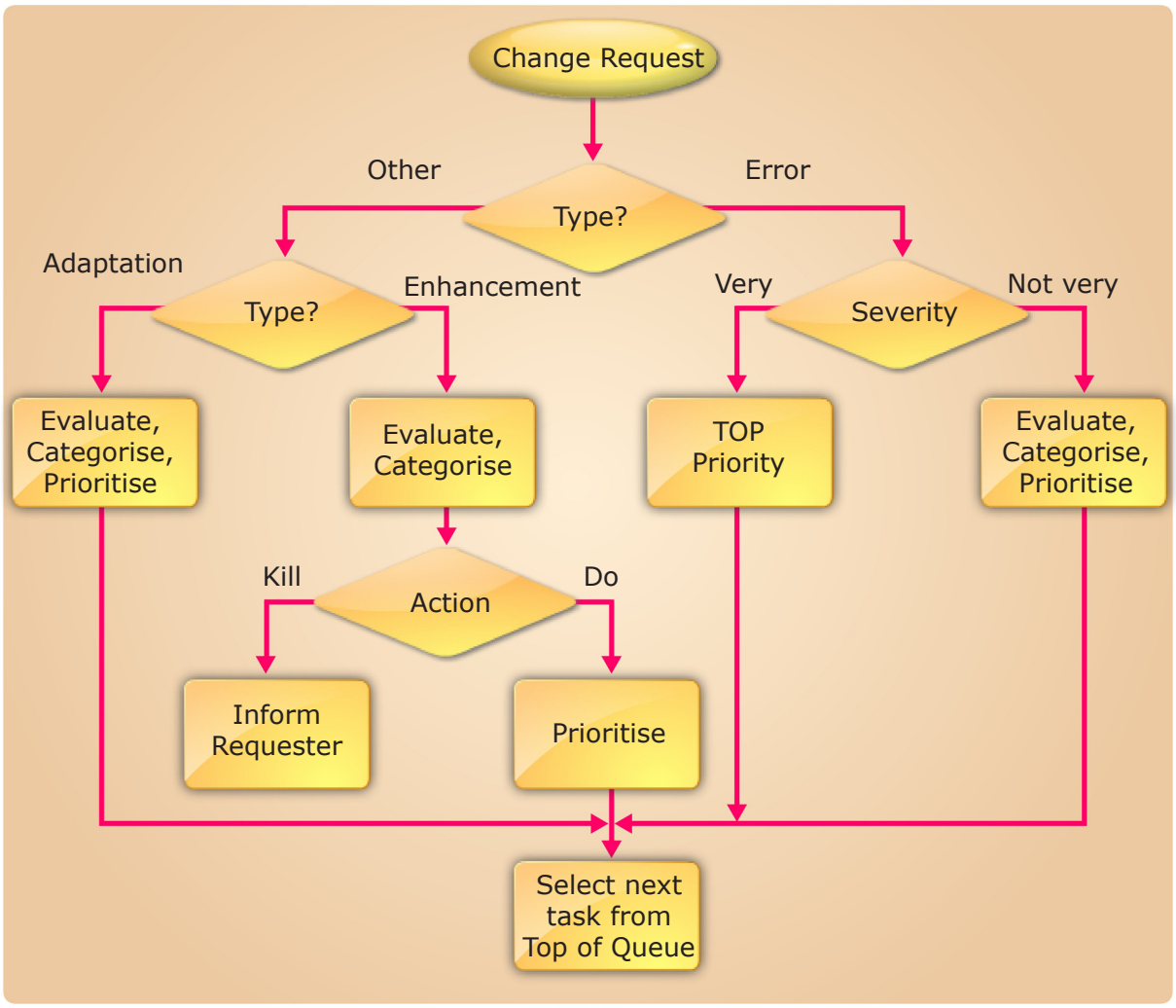


Figure 10.3: Flowchart of how to control maintenance requests
Source: Adapted from Pressman (2005)

Managing the queue of maintenance tasks also another important activity. The queue can be growing and shrinking based upon business changes and errors. To have the overview of change request flow, see Figure 10.3.

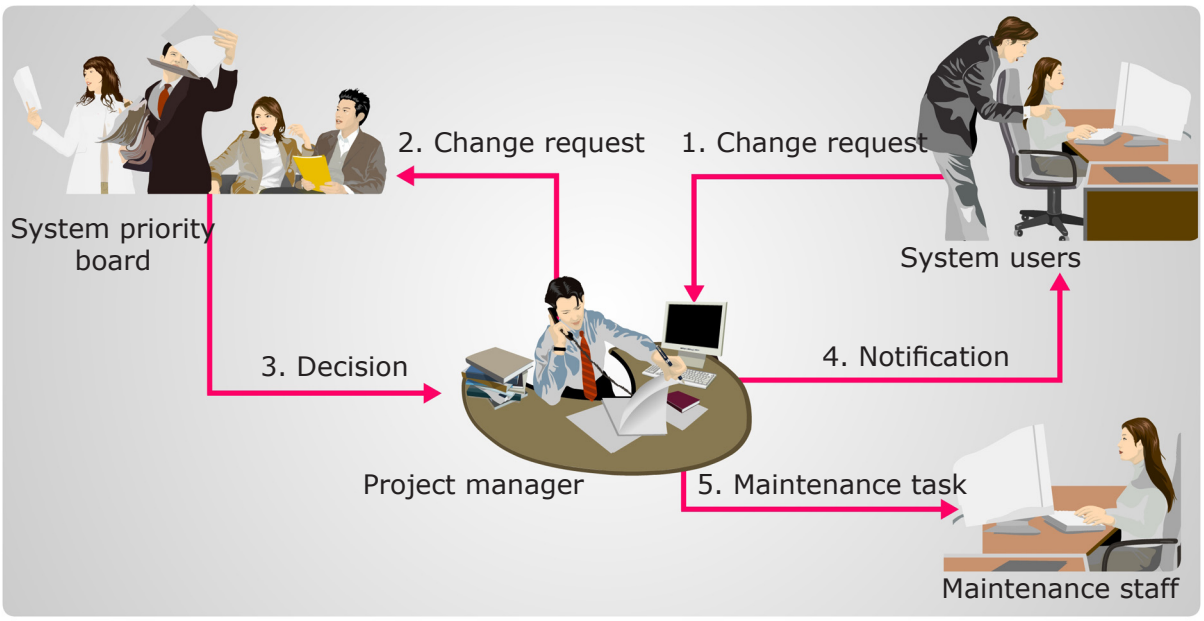


Figure 10.4: How a maintenance request moves through an organisation
Source: Adapted from Hoffer et al (2008)

Although each change request goes through the approval process depicted in Figure 10.4, changes are usually implemented in batches, forming a new release of the software. It is too difficult to manage a lot of small changes. Frequent releases of new system versions may also confuse users if the appearance of displays, reports or data entry screens changes.

ACTIVITY

What are the different types of maintenance and how do they differ?

SUMMARY

- This chapter presented an overview of the various task of the system implementation phase. It consists of coding, testing, installation, documentation and training. You learned several different types of testing until the system is able to meet user requirement. You learned about four different types of installation and strategy to select the appropriate installation types. In addition, this chapter also describes four different types of documentation and several choices of training to help user to understand the system and know how to use it effectively.
- How the system is designed and implemented can greatly impact in system maintenance performance. During maintenance, systems are changed to rectify errors or to extend the functionality of the system. This chapter describes several different types of maintenances request and managing effective maintenance activities which need for careful management. Appropriate understanding of the system development cycle will reduce expenditure in system maintenance and improve the delivery of the system implementation.

KEY TERMS

Acceptance Testing

Adaptive Maintenance

Alpha Testing

Beta Testing

Code Review

Corrective Maintenance

Desk Checking

Design Walkthrough

Direct Installation

Documentation

Integration Testing

Link Testing

Mean Time Between Failures (MTBF)

Module Testing

Operations Documentation

Parallel Installation

Perfective Maintenance

Phased Installation

Pilot Installation

Preventive Maintenance

Program Documentation

Structured Walkthrough

Stub Testing

System Documentation

System Implementation

System Maintenance

System Testing

Syntax Errors

Test Data

Unit Testing

User Documentation

REFERENCES

1. Bell, P., and C.Evans. 1989. Mastering Documentation. New York: John Wiley & Sons
2. Eason, K.1998. Information Technology and Organisational Change. London: Taylor & Francis
3. Ginzberg. M.J. 1981b. "Key Recurrent Issues in the MIS Implementation Process" MIS Quarterly 5 (2): 47-59
4. Hoffer, J.A., George, J.F. and Valacich (2008) 5th ed., Modern Systems Analysis and Design, Benjamin/Cummings, Massachusetts.
5. Ives, B., and M.H. Olson. 1984. "User Involvement and MIS Success: A Review of Research"Management Science 30 (5):586-603