CHAPTER 8

System Implementation and Maintenance

- ❏ Implementation is a process of ensuring that the information system is operational.
- ❏ It involves:
  - ❖ Constructing a new system from scratch
  - ❖ Constructing a new system from the existing one
- ❏ Implementation allows the users to take over its operation for use and evaluation.
- ❏ It involves training the users to handle the system and plan for a smooth conversion

8.1 Conversion

- It is a process of migrating from the old system to the new one.
- It provides understandable and structured approach to improve the communication between management and project team.

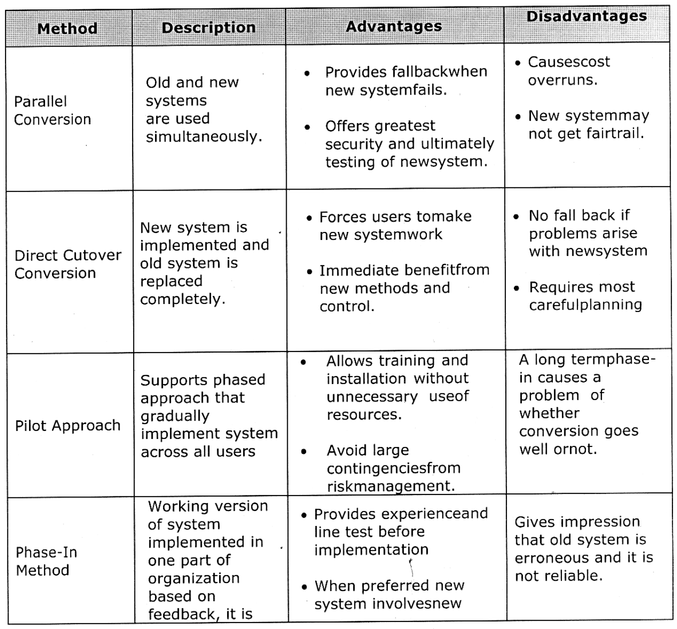
8.1.1 Conversion Plan

A conversion plan is the design to change over from the existing system to the new system. A properly designed conversion plan ensures a smooth transition to the new system. Database is designed, created and installed by using the existing data from the old system or by creating data manually. Appropriate training of the end user is important as the success of any system depends on the involvement of end user who is actually going to use the system

It contains description of all the activities that must occur during implementation of the new system and put it into operation. It anticipates possible problems and solutions to deal with them.

It includes the following activities:

- Name all files for conversions.
- Identifying the data requirements to develop new files during conversion.
- Listing all the new documents and procedures that are required.
- Identifying the controls to be used in each activity.
- Identifying the responsibility of person for each activity.
- Verifying conversion schedules.



Method	Description	Advantages	Disadvantages
Parallel Conversion	Old and new systems are used simultaneously.	• Provides fallbackwhen new systemfails. • Offers greatest security and ultimately testing of newsystem.	• Causescost overruns. • New systemmay not get fairtrail.
Direct Cutover Conversion	New system is implemented and old system is replaced completely.	• Forces users tomake new systemwork • Immediate benefitfrom new methods and control.	• No fall back if problems arise with newsystem • Requires most carefulplanning
Pilot Approach	Supports phased approach that gradually implement system across all users	• Allows training and installation without unnecessary useof resources. • Avoid large contingenciesfrom riskmanagement.	A long termphase-in causes a problem of whether conversion goes well ornot.
Phase-In Method	Working version of system implemented in one part of organization based on feedback, it is	• Provides experienceand line test before implementation • When preferred new system involvesnew	Gives impression that old system is erroneous and it is not reliable.

Figure 8.1: Conversion/installation methods

8.1.2 Conversion/Installation Methods

- ❏ The conversion style is the way users are switched between the old and new systems.
- ❏ The *four* methods of conversion/installation are:

Conversion/Installation method

1. Direct conversion	3. Pilot approach
2. Parallel conversion	4. Phase-in method

8.1.2.1 Direct Conversion

❏ With direct conversion (sometimes called cold turkey, big bang, 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 we are likely to use when we upgrade commercial software (e.g., Microsoft Word) from one version to another; we simply begin using the new version and stop using the old version.

❏ Direct conversion 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 can seriously disrupt the organization.

8.1.2.2 Parallel Conversion

❏ With parallel conversion, 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 organization 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 organization continues using the new system.

❏ This approach is more likely to catch any major bugs in the new system and prevent the organization from suffering major problems.

❏ If problems are discovered in the new system, the system is simply turned off and fixed and then the conversion process starts again.

❏ The problem with this approach is the added expense of operating two systems that perform the same function.

8.1.2.3 Pilot Conversion

❏ With a pilot conversion, one or more locations or units or work groups within a location are selected to be converted first as part of a pilot test.

❏ The locations participating in the pilot test are converted (using either direct or parallel conversion).

❏ If the system passes the pilot test, then the system is installed at the remaining locations (again using either direct or parallel conversion).

❏ Pilot conversion has the advantage of providing an additional level of testing before the system is widely deployed throughout the organization, so that any problems with the system affect only the pilot locations.

❏ However, this type of conversion obviously requires more time before the system is installed at all organizational locations.

❏ Also, it means that different organizational units are using different versions of the system and business processes, which can make it difficult for them to exchange data.

8.1.2.4 Phased Conversion

❏ With phased conversion, the system is installed sequentially at different locations.

❏ A first set of locations is converted, then a second set, then a third set, and so on, until all locations are converted. 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 organization is affected.

❏ In other cases, the sets are converted back to back so that as soon as those converting one location have finished, the project team moves to the next and continues the conversion.

❏ Phased conversion has the same advantages and disadvantages of pilot conversion.

❏ In addition, it means that fewer people are required to perform the actual conversion (and any associated user training) than if all locations were converted at once.

8.2 System Maintenance

❏ Maintenance means restoring something to its original conditions.

❏ Enhancement means adding, modifying the code to support the changes in the user specification.

❏ System maintenance conforms the system to its original requirements and enhancement adds to system capability by incorporating new requirements.

- ❏ Thus, maintenance changes the existing system, enhancement adds features to the existing system, and development replaces the existing system.
- ❏ It is an important part of system development that includes the activities which corrects errors in system design and implementation, updates the documents, and tests the data.
- ❏ Any maintenance activity comprises the following *four* key stages:

Help Desk A preliminary analysis of the change request submitted by the user through a formal change request will be done, and if the problem is sensible, it is accepted.

Analysis Managerial and technical analysis of the problems are undertaken to investigate the cost factors and other alternative solutions.

Implementation The maintenance team implements the chosen change/ solution as well as tested by them. All infected components are to be identified and brought in to the scope of the change. Unit test, integration test, user-oriented functional acceptance tests and regression test strategies are provided.

Release The changes are released to the customer, with a release note and appropriate documentation giving details of the changes

8.2.1 Types of Maintenance

- ❏ Once the system is fully implemented and starts operating, the maintenance phase begins.

- The following are the different types of maintenance activities:

Types of maintenance

1. Corrective maintenance
2. Adaptive maintenance
3. Perfective maintenance
4. Preventive maintenance

8.2.1.1 Corrective Maintenance

- This type of maintenance is to rectify design, coding and implementation problems detected after the implementation of the System.
- This kind of problem generally surfaces immediately after the system is implemented.
- This type of problem needs immediate attention as it hampers the day to day work of the end user.
- Proper planning and interaction with the end user during system development process can minimize Corrective Maintenance.
- In spite of the all these kinds of maintenance, these constitute more than 60 percent of total maintenance effort.
- Corrective maintenance is very much undesirable.
- Care should be taken to see that normal business operations are not disturbed because of it.

8.2.1.2 Adaptive Maintenance

- ❏ Adaptive maintenance is required because business operates on a social environment and need of the organization changes as organization ventures in to new areas, or as government regulation policy changes, etc.
- ❏ Maintenance of the software to adapt to this kind of changes is called adaptive maintenance.
- ❏ Unlike corrective maintenance, this kind of activity adds value to the information system and affects a small part of the organization.
- ❏ This activity is not as urgent as corrective maintenance as these changes are gradual and allow sufficient time to the system group to make changes to the software.

8.2.1.3 Perfective Maintenance

- ❏ Adding new functionalities and features to the software to make it more versatile and user oriented is the characteristics of Perfective Maintenance.
- ❏ Some times, changes are made to improve performance of the software.
- ❏ In some sense, this maintenance can be thought of as a new development activity.

8.2.1.4 Preventive Maintenance

- ❏ The activity done to prevent system failure by bringing changes to software for the system safe future and easy maintenance comes under preventive maintenance.

- ❏ This reduces the need of corrective maintenance.
- ❏ Preventive maintenance could increase the volume of transactions that can be handled by the system. Preventive maintenance is done when the system is least used or not used at all.
- ❏ This does not add value to the system, but certainly lowers the cost of corrective maintenance.

Issues Involved in Maintenance

The responsibility of the software development team and clients does not end once the product is released for implementation and installed. It is very important that the Software should be easily maintainable. Factors like availability of source code, availability of system manuals, etc., are very important for maintainability. One of the most important issues is the cost factor for maintenance of software.

The following are various factors which affect the ease of maintenance:

Volume of Defects The inherent errors / bugs that are found in the system after installation. Cost of maintenance increases with the increase in volume of defects.

Number of Customers More number of customers means more requests for changes in the system after installation.

Availability of System Documentation The quality and availability of system documentation is vital to carry out the maintenance. Poorly written system documentation increases the cost of maintenance. Most often, the programmers for development are different than the team of programmers for maintenance and the later often finds it difficult to understand a program written by the former. Structured programming and program documentations are very useful in maintaining the system

The following are various issues in Software Maintenance:

Organizational Issues

❏ The maintenance activity must align with organizational objectives.

❏ Most of the Software Maintenance activity is resource consuming and it has no clear quantifiable benefit for the organization.

❏ Outsourcing the job of Software Maintenance

Process Issues

❏ Software Maintenance requires a number of additional activities not performed at development stage

❏ Impact analysis and Regression tests on the software changes are crucial issues

Technical Issues

❏ How to construct software that it is easy to comprehend is a major issue and the technology to do this is still not available.

Legacy System

❏ A legacy system is typically a very old and large system which has been modified heavily since it started operation.

- Legacy systems are based on old technology with very little or no documentation.
- Dealing with a legacy system can be very hard.

8.3 Software Quality Assurance

Software Quality

- Quality, simplistically, means that a product should meet its specification.
- This is problematical for software systems
  - There is a tension between customer quality requirements (efficiency, reliability, etc.) and developer quality requirements (maintainability, reusability, etc.);
  - Some quality requirements are difficult to specify in an unambiguous way;
  - Software specifications are usually incomplete and often inconsistent.
- The focus may be 'fitness for purpose' rather than specification conformance.
- There are two kinds of quality
  1. *Quality of Design*
     - Quality of Design refers to the characteristics that designers specify for an item.
     - The grade of materials, tolerances, and performance specifications that all contribute to the quality of design.

2. *Quality of conformance*
    - ❖ Quality of conformance is the degree to which the design specifications are followed during manufacturing.
    - ❖ Greater the degree of conformance, the higher is the level of quality of conformance.

Quality Assurance

- ❏ Quality Assurance is defined as a procedure to ensure the quality of software products or services provided to the customers by an organization.
- ❏ Quality assurance focuses on improving the software development process and making it efficient and effective as per the quality standards defined for software products.
- ❏ Quality Assurance is popularly known as *QA Testing.*

SQA Activities

- ❏ Software quality assurance is composed of a variety of functions associated with two different constituencies
    - ❖ the software engineers who do technical work and
    - ❖ an SQA group that has responsibility for quality assurance planning, record keeping, analysis, and reporting.
- ❏ SQA activities involves :
    - ❖ Preparation of SQA plan for project
    - ❖ Participates in the development of the project's software process description and applying the software engineering techniques for development.

- ❖ Reviews software engineering activities like Formal Technical Review (FTR) to verify compliance with the defined software process
- ❖ Applying a multi-tiered testing strategy
- ❖ Controlling change and measuring the impact of change
- ❖ Audits designated software work products to verify compliance with those defined as a part of the software process
- ❖ Ensures that deviations in software work and work products are documented and handled according to a documented procedure
- ❖ Records any noncompliance and reports to senior management

Benefits of Software Quality Assurance (SQA)

- ❏ SQA produce high quality software.
- ❏ High quality application saves time and cost.
- ❏ SQA is beneficial for better reliability.
- ❏ SQA is beneficial in the condition of no maintenance for long time.
- ❏ High quality commercial software increase market share of company.
- ❏ Improving the process of creating software.
- ❏ Improves the quality of the software.

Quality Assurance

❏ Quality Assurance is defined as a procedure to ensure the quality of software products or services provided to the customers by an organization.

❏ Quality assurance focuses on improving the software development process and making it efficient and effective as per the quality standards defined for software products.

❏ Quality Assurance is popularly known as QA Testing.

❏ Software quality assurance (SQA) is a process which assures that all software engineering processes, methods, activities and work items are monitored and comply against the defined standards. These defined standards could be one or a combination of any like ISO 9000, CMMI model, ISO15504, etc

❏ This stage can include:

   ❖ encouraging documentation process standards, such as the creation of well-defined engineering documents using standard templates

   ❖ mentoring how to conduct standard processes, such as quality reviews

   ❖ performing in-process test data recording procedures

   ❖ identifying standards, if any, that should be used in software development processes

Quality Control

❏ Quality control popularly abbreviated as QC. It is a Software Engineering process used to ensure quality in a product or a service.

- It does not deal with the processes used to create a product; rather it examines the quality of the "end products" and the final outcome.

- The main aim of Quality control is to check whether the products meet the specifications and requirements of the customer. If an issue or problem is identified, it needs to be fixed before delivery to the customer.

- Walkthrough, Testing, inspection, checkpoint review are the example of quality control.

- Activities include:

  - release testing of software, including proper documentation of the testing process
  - examination of software and associated documentation for non-conformance with standards
  - follow-up review of software to ensure any required changes detailed in previous testing are addressed
  - application of software measurement and metrics for assessment

SQA Techniques

Auditing Auditing involves inspection of the work products and its related information to determine if the set of standard processes were followed or not.

Reviewing A meeting in which the software product is examined by both the internal and external stakeholders to seek their comments and approval.

Code Inspection It is the most formal kind of review that does static testing to find bugs and avoid defect growth in the later stages. It is done by a trained mediator/peer and is based on rules, checklist, entry and exit criteria. The reviewer should not be the author of the code.

Design Inspection Design inspection is done using a checklist that inspects the below areas of software design:

- ❏ General requirements and design
- ❏ Functional and Interface specifications
- ❏ Conventions
- ❏ Requirement traceability
- ❏ Structures and interfaces
- ❏ Logic
- ❏ Performance
- ❏ Error handling and recovery
- ❏ Testability, extensibility
- ❏ Coupling and cohesion

Simulation Simulation is a tool that models the real-life situation in order to virtually examine the behavior of the system under study.

Functional Testing It is a QA technique which verifies what the system does without considering how it does. This type of black box testing mainly focuses on testing the system specifications or features.

Standardization Standardization plays a crucial role in quality assurance. It decreases the ambiguity and guesswork, thus ensuring quality.

Static Analysis It is a software analysis that is done by an automated tool without actually executing the program. This technique is highly used for quality assurance in medical, nuclear and aviation software. Software metrics and reverse engineering are some popular forms of static analysis.

Walkthroughs Software walkthrough or code walkthrough is a kind of peer review where the developer guides the members of the development team to go through the product and raise queries, suggest alternatives, make comments regarding possible errors, standard violations or any other issues.

Path Testing It is a white box testing technique where the complete branch coverage is ensured by executing each independent path at least once.

Stress Testing This type of testing is done to check how robust a system is by testing it under heavy load i.e. beyond normal conditions.

Six Sigma Six Sigma is a quality assurance approach that aims at nearly perfect products or services. It is widely applied in many fields including software. The main objective of six sigma is process improvement so that the produced software is 99.76 % defect free.

8.3.1 Software Review

❏ Software Review is systematic inspection of a software by one or more individuals who work together to find and resolve errors and defects in the software during the early stages of Software Development Life Cycle (SDLC).

- ❏ Software review is an essential part of Software Development Life Cycle (SDLC) that helps software engineers in validating the quality, functionality and other vital features and components of the software.
- ❏ It is a whole process that includes testing the software product and it makes sure that it meets the requirements stated by the client.
- ❏ Usually performed manually, software review is used to verify various documents like requirements, system designs, codes, test plans and test cases.

Objectives of Software Review

Objectives of Software Review (why software review is important?) The objective of software review is:

- ❏ To improve the productivity of the development team.
- ❏ To make the testing process time and cost effective.
- ❏ To make the final software with fewer defects.
- ❏ To eliminate the inadequacies.
- ❏ To train technical authors for defect detection process as well as for "defect prevention process".
- ❏ To reduce the time taken in producing a technically sound document.

8.3.2 Techniques of Review

Techniques of review

1. Inspection
2. Walkthrough
3. Formal technical reviews (FTR)

8.3.2.1 Inspection

❏ The term software inspection was developed by IBM in the early 1970s, when it was noticed that the testing was not enough sufficient to attain high quality software for large applications.

❏ Inspection is used to determine the defects in the code and remove it efficiently.

❏ This prevents defects and enhances the quality of testing to remove defects.

❏ This software inspection method achieved the highest level for efficiently removing defects and improving software quality.

❏ Software Inspections refers to peer review of any work product by trained individuals who look for defects using a well defined process.

  ❖ It is usually manual and a static technique that is applied in the early development cycle.

  ❖ Software inspection is regarded as the most formal type of review.

- ❖ It is led by the trained moderators and involves peers to examine the product.
- ❖ The defects found during this process are documented in a issue log for convenience.

Software Inspection Process The process must have an entry criterion that determines whether the inspection process is ready to begin. this prevents incomplete products from entering the inspection process. Entry criteria can be interstitial with items such as "The Spell-Document Check". There are some of the stages in the software inspection process such as-

Planning The moderator plan the inspection.

Overview Meeting The background of the work product is described by the author.

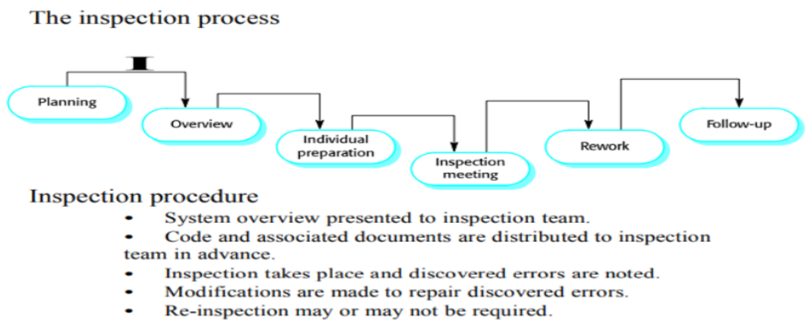
Preparation The examination of the work product is done by inspector to identify the possible defects.

Inspection Meeting The reader reads the work product part by part during this meeting and the inspectors the faults of each part.

Rework after the inspection meeting, the writer changes the work product according to the work plans.

Follow Up The changes done by the author are checked to make sure that everything is correct.

Role of Inspection Team An inspection team consists of three to eight members who play the roles of moderator, author, reader, recorder, and inspector.



The inspection process

Inspection procedure

- System overview presented to inspection team.
- Code and associated documents are distributed to inspection team in advance.
- Inspection takes place and discovered errors are noted.
- Modifications are made to repair discovered errors.
- Re-inspection may or may not be required.

Figure 8.2: Inspection process

Moderator leads the inspection, schedules meetings, controls meetings, reports inspection results, and follows up on rework issues.

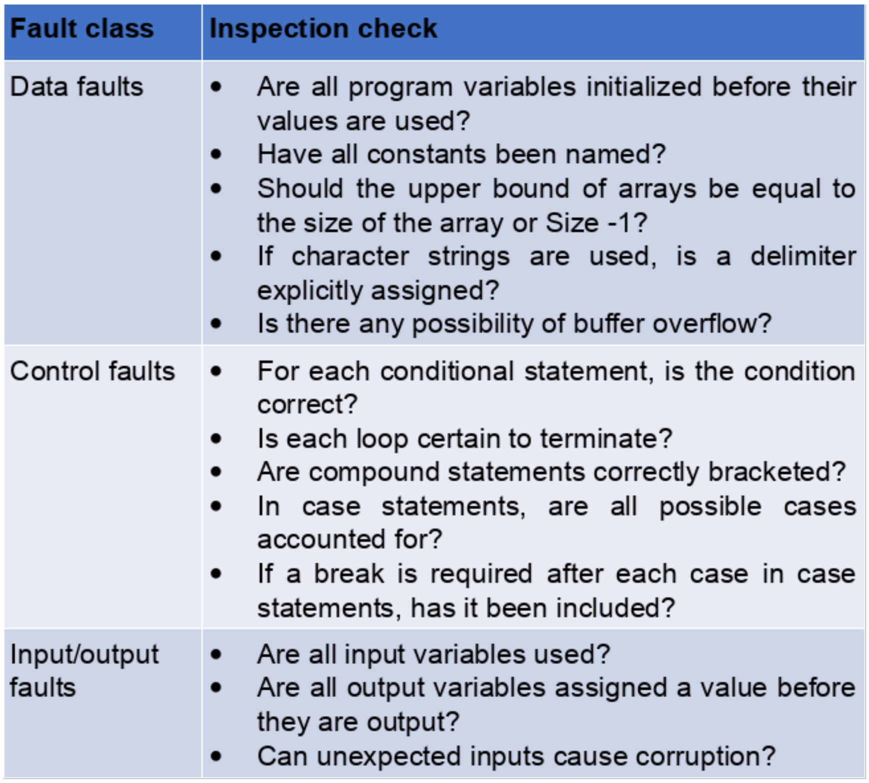
Author creates or maintains the work product being inspected.

Reader describes the sections of the work product to the team as they proceed through inspection.

Recorder classifies and records defects and issues raised during the inspection. The moderator might perform this role in a small inspection team.

Inspector finds errors in the product. All participants play the role of inspectors. However, good inspectors are those who have created the specification for the work product being inspected.

- ❏ For example, the designer can act as an inspector during code inspection while a quality assurance representative can act as standard enforcer. For example, the designer can act as an inspector during code inspection while a quality assurance representative can act as standard enforcer. It also helps to have a client representative participate in requirements specification inspections.



Fault class	Inspection check
Data faults	• Are all program variables initialized before their values are used? • Have all constants been named? • Should the upper bound of arrays be equal to the size of the array or Size -1? • If character strings are used, is a delimiter explicitly assigned? • Is there any possibility of buffer overflow?
Control faults	• For each conditional statement, is the condition correct? • Is each loop certain to terminate? • Are compound statements correctly bracketed? • In case statements, are all possible cases accounted for? • If a break is required after each case in case statements, has it been included?
Input/output faults	• Are all input variables used? • Are all output variables assigned a value before they are output? • Can unexpected inputs cause corruption?

Figure 8.3: Inspection checklist

8.3.2.2 Walkthroughs

- ❏ The term walkthrough refers to a group activity in which the developer of the product guides the progress of the review.
- ❏ Walkthroughs are less rigorous than either formal inspections or peer reviews in which the developer plays a more passive role.
- ❏ Normally walkthroughs turn into a presentation by the author.
- ❏ The focus of finding errors is diluted. Such misadventures make walkthroughs usually less successful at detecting bugs than the

more formal review methods.

❏ Two useful walkthrough approaches adopted worldwide are:

1. Group reviews with individual preparation and
2. individual peer desk-checks.

Group reviews are not very rigorous like inspections. The group reviews involve many of the same activities and roles, such as individual preparation and the use of a moderator and a recorder. Usually the overview meeting and the follow-up steps are skipped and checklists are used sparingly. At the end, the readers paraphrase their interpretation of what a program is doing.

Individual peer desk-checks are quite cost-effective. Only one person besides the author examines the material. This approach can be more effective if there are individuals who are extremely good at finding defects on their own. If someone consistently finds most of the group-identified defects during the individual preparation step, such a person is fit to perform individual peer desk-checks.

8.3.2.3 Formal Technical Reviews (FTR)

❏ Formal Technical Review (FTR) is a software quality control activity performed by software engineers.

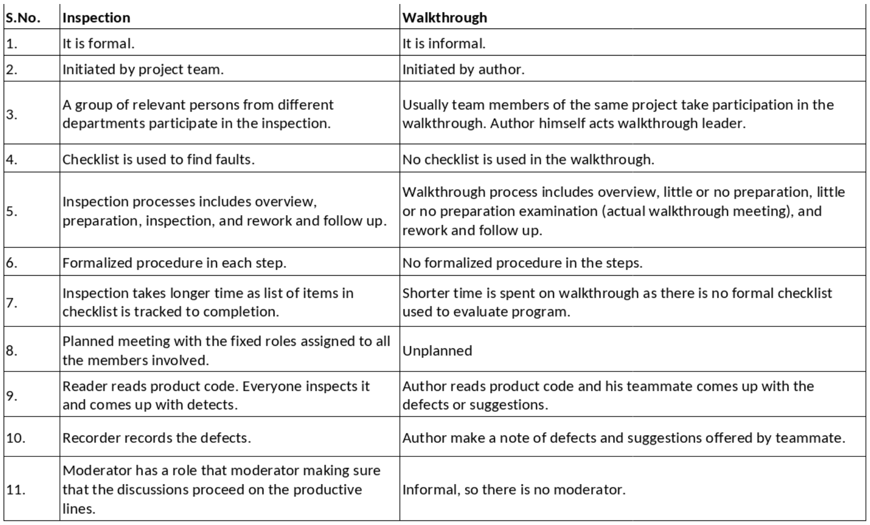
❏ This activity is performed to check errors in logic, function, or implementation for any representation of the software.

❏ Using FTR, you can verify whether or not the software product adheres to the defined standards.

- ❏ FTR is also conducted to check for consistency in the software product and audit the manageability of the software product.
- ❏ It includes activities such as walkthrough and inspection.
- ❏ FTR focuses on specific parts of the software product, such as the requirements components, detailed design module, and the source code listing for a module.
- ❏ FTR also concentrates on the entire product. The participants in FTR are the developer, the project leader, and all the reviewers.
- ❏ At the end of the review meeting, the issues recorded are formalized into review issues list.

Objectives of FTR

- ❏ Useful to uncover error in logic, function and implementation for any representation of the software.
- ❏ The purpose of FTR is to verify that the software meets specified requirements.
- ❏ To ensure that software is represented according to predefined standards.
- ❏ It helps to review the uniformity in software that is development in a uniform manner.
- ❏ To makes the project more manageable.



S.No.	Inspection	Walkthrough
1.	It is formal.	It is informal.
2.	Initiated by project team.	Initiated by author.
3.	A group of relevant persons from different departments participate in the inspection.	Usually team members of the same project take participation in the walkthrough. Author himself acts walkthrough leader.
4.	Checklist is used to find faults.	No checklist is used in the walkthrough.
5.	Inspection processes includes overview, preparation, inspection, and rework and follow up.	Walkthrough process includes overview, little or no preparation, little or no preparation examination (actual walkthrough meeting), and rework and follow up.
6.	Formalized procedure in each step.	No formalized procedure in the steps.
7.	Inspection takes longer time as list of items in checklist is tracked to completion.	Shorter time is spent on walkthrough as there is no formal checklist used to evaluate program.
8.	Planned meeting with the fixed roles assigned to all the members involved.	Unplanned
9.	Reader reads product code. Everyone inspects it and comes up with detects.	Author reads product code and his teammate comes up with the defects or suggestions.
10.	Recorder records the defects.	Author make a note of defects and suggestions offered by teammate.
11.	Moderator has a role that moderator making sure that the discussions proceed on the productive lines.	Informal, so there is no moderator.

Figure 8.4: Inspection vs walkthrough

8.4 Test Planning

- Testing starts with the development of a test plan, which defines a series of tests that will be conducted.
- Because testing takes place throughout the development of an object-oriented system, a test plan should be developed at the very beginning of system development and continuously updated as the system evolves.

8.4.1 Unit Tests

- Unit tests focus on one unit –a program module that performs a specific function that can be tested.
- The purpose of a unit test is to ensure that the module or program performs its function as defined in the program specification.

- ❏ Unit testing is performed after the programmer has developed and tested the code and believes it to be error free.
- ❏ These tests are based strictly on the program specification and may discover erros resulting from the programmer's misinterpretation of the specifications.
- ❏ Unit tests are often conducted by the system analyst or, sometimes, by the programmers who developed the unit.
- ❏ Two approaches to unit testing:
    1. black box testing
        - ❖ Black-box testing is the most commonly used because each class represents an encapsulated object.
        - ❖ Black-box testing is driven by behavior state machines, and contracts associated with a class, not by the programmers' interpretation.
        - ❖ In this case, the test plan is developed directly from the spec- ification of the class: each item in the specification becomes a test, and several test cases are developed for it.
    2. white-box testing
        - ❖ White-box testing is based on the method specifications associated with each class.
        - ❖ However, white-box testing has had limited impact in object-oriented development. This is due to the rather small size of the individual methods in a class.
        - ❖ Most approaches to testing classes use black-box testing to ensure their correctness.

8.4.2 Integration Tests

❏ Integration tests assess whether a set of classes that must work together do so without error.

❏ They ensure that the interfaces and linkages between different parts of the system work properly.

❏ At this point, the classes have passed their individual unit tests, so the focus now is on the flow of control among the classes and on the data exchanged among them.

❏ Integration testing follows the same general procedures as unit testing: The tester develops a test plan that has a series of tests, which, in turn, have a test. Integration testing is often done by a set of programmers and/or systems analysts.

❏ There are *four approaches to integration testing*: user interface testing,

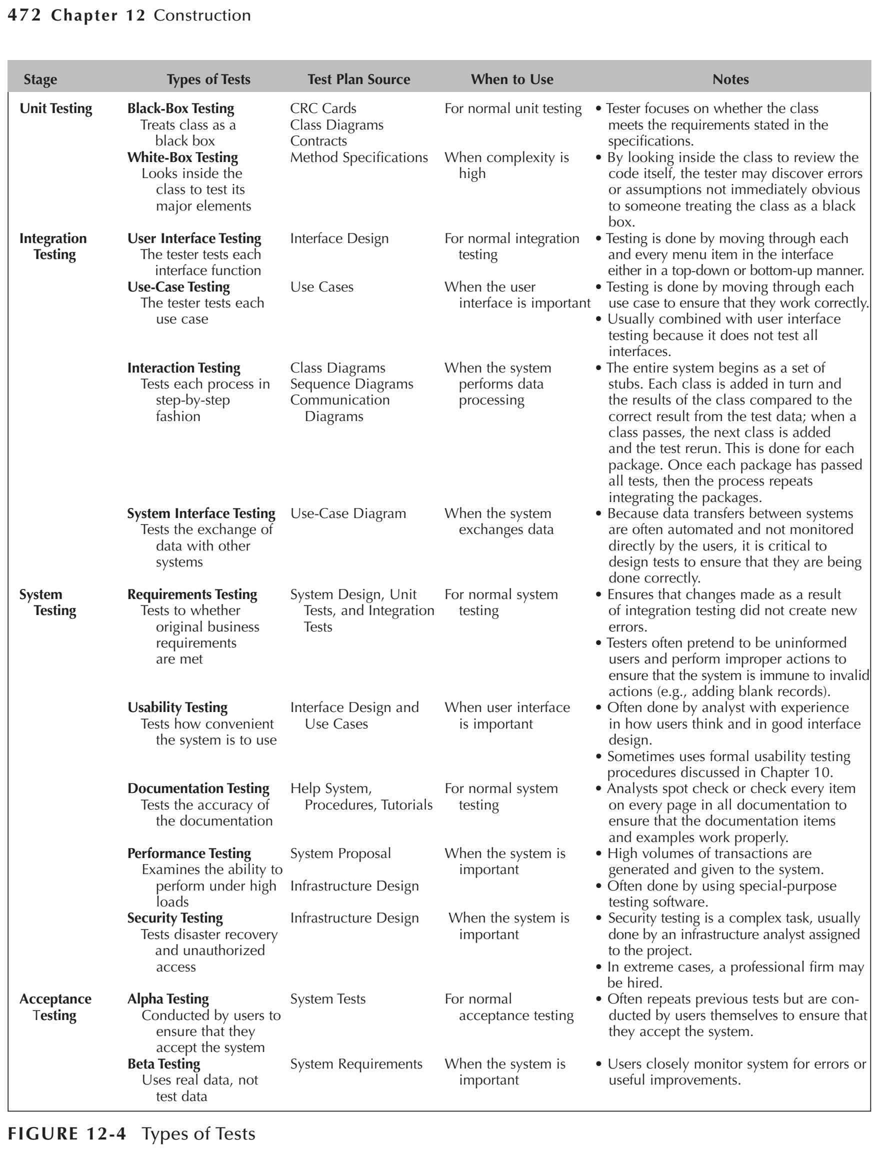
1. user interface testing
2. use-case testing,
3. interaction testing, and
4. system interface testing

❏ Most projects use all four approaches.

8.4.3 System Tests

❏ To ensure that all classes work together without error, systems analysts usually conduct the system tests.

❏ System testing is similar to integration testing but is much broader in scope.



472 Chapter 12 Construction

Stage	Types of Tests	Test Plan Source	When to Use	Notes
Unit Testing	**Black-Box Testing** Treats class as a black box	CRC Cards Class Diagrams Contracts	For normal unit testing	• Tester focuses on whether the class meets the requirements stated in the specifications.
	White-Box Testing Looks inside the class to test its major elements	Method Specifications	When complexity is high	• By looking inside the class to review the code itself, the tester may discover errors or assumptions not immediately obvious to someone treating the class as a black box.
Integration Testing	**User Interface Testing** The tester tests each interface function	Interface Design	For normal integration testing	• Testing is done by moving through each and every menu item in the interface either in a top-down or bottom-up manner.
	Use-Case Testing The tester tests each use case	Use Cases	When the user interface is important	• Testing is done by moving through each use case to ensure that they work correctly. • Usually combined with user interface testing because it does not test all interfaces.
	Interaction Testing Tests each process in step-by-step fashion	Class Diagrams Sequence Diagrams Communication Diagrams	When the system performs data processing	• The entire system begins as a set of stubs. Each class is added in turn and the results of the class compared to the correct result from the test data; when a class passes, the next class is added and the test rerun. This is done for each package. Once each package has passed all tests, then the process repeats integrating the packages.
	System Interface Testing Tests the exchange of data with other systems	Use-Case Diagram	When the system exchanges data	• Because data transfers between systems are often automated and not monitored directly by the users, it is critical to design tests to ensure that they are being done correctly.
System Testing	**Requirements Testing** Tests to whether original business requirements are met	System Design, Unit Tests, and Integration Tests	For normal system testing	• Ensures that changes made as a result of integration testing did not create new errors. • Testers often pretend to be uninformed users and perform improper actions to ensure that the system is immune to invalid actions (e.g., adding blank records).
	Usability Testing Tests how convenient the system is to use	Interface Design and Use Cases	When user interface is important	• Often done by analyst with experience in how users think and in good interface design. • Sometimes uses formal usability testing procedures discussed in Chapter 10.
	Documentation Testing Tests the accuracy of the documentation	Help System, Procedures, Tutorials	For normal system testing	• Analysts spot check or check every item on every page in all documentation to ensure that the documentation items and examples work properly.
	Performance Testing Examines the ability to perform under high loads	System Proposal Infrastructure Design	When the system is important	• High volumes of transactions are generated and given to the system. • Often done by using special-purpose testing software.
	Security Testing Tests disaster recovery and unauthorized access	Infrastructure Design	When the system is important	• Security testing is a complex task, usually done by an infrastructure analyst assigned to the project. • In extreme cases, a professional firm may be hired.
Acceptance Testing	**Alpha Testing** Conducted by users to ensure that they accept the system	System Tests	For normal acceptance testing	• Often repeats previous tests but are conducted by users themselves to ensure that they accept the system.
	Beta Testing Uses real data, not test data	System Requirements	When the system is important	• Users closely monitor system for errors or useful improvements.

FIGURE 12-4 Types of Tests

Figure 8.5: Types of tests

❏ Whereas integration testing focuses on whether the classes work together without error, system tests examine how well the system meets both the functional and nonfunctional requirements, e.g., usability, documentation, performance, and security.

8.4.4 Acceptance Tests

❏ Acceptance testing is done primarily by the users with support from the project team.

❏ The goal is to confirm that the system is complete, meets the business needs that prompted the system to be developed, and is acceptable to the users.

❏ Acceptance testing is done in two stages:

1. *alpha testing*, in which users test the system using made-up data, and
2. *beta testing,* in which users begin to use the system with real data but are carefully monitored for errors

8.4.5 Process of Maintenance

8.4.5.1 Issue Identification and Reporting

❏ The first step in system maintenance is identifying issues or problems that need attention.

❏ These issues can arise from user feedback, system monitoring, error logs, or other sources. Users or stakeholders report these issues to the maintenance team or help desk.

8.4.5.2 Issue Classification and Prioritization

❏ Once issues are identified, they need to be classified and prioritized based on their impact, urgency, and severity.

❏ The maintenance team assesses each issue, categorizes it, and determines its priority level.

❏ Critical or high-impact issues may require immediate attention, while minor issues can be addressed later.

8.4.5.3 Issue Analysis and Root Cause Identification

❏ The maintenance team analyzes each reported issue to understand its root cause.

❏ They investigate the system behavior, examine error logs, review code, and perform other diagnostic activities to identify the underlying cause of the issue.

❏ This step helps in understanding the problem in-depth and formulating an appropriate solution.

8.4.5.4 Solution Development and Testing:

❏ Once the root cause is identified, the maintenance team develops a solution to address the issue.

❏ This may involve modifying code, fixing configuration settings, applying patches, or making other changes to the system.

❏ The solution is then thoroughly tested to ensure that it resolves the issue without introducing new problems.

❏ After testing the developed solution it is deployed in the system.

8.4.5.5 Monitoring and Performance Evaluation:

- ❏ Once the system is back in operation, it is continuously monitored to ensure its performance, stability, and reliability.
- ❏ Monitoring tools and techniques are employed to track system behavior, detect anomalies, and proactively address potential issues.
- ❏ Performance evaluations are conducted to assess the system's performance against predefined metrics and identify areas for improvement.

8.4.5.6 Regular Maintenance Activities:

- ❏ System maintenance also involves regular activities, such as backups, security updates, software upgrades, and periodic reviews of the system's performance and stability.
- ❏ These routine maintenance tasks help ensure the system's long-term health and stability.

8.4.5.7 Continuous Improvement

- ❏ System maintenance is an iterative process, and continuous improvement is an essential aspect.
- ❏ Lessons learned from past maintenance activities, user feedback, and system performance data are analyzed to identify areas for improvement.
- ❏ Process refinements, automation, and other measures are implemented to enhance the effectiveness and e fficiency of the maintenance process.

8.5 System Testing

- ❏ To ensure that all classes work together without error, systems analysts usually conduct the system tests.
- ❏ System testing is similar to integration testing but is much broader in scope.
- ❏ Whereas integration testing focuses on whether the classes work together without error, system tests examine how well the system meets both the functional and nonfunctional requirements, e.g., usability, documentation, performance, and security.

Functional requirements testing ensure that the functional require ments uncovered are indeed met. Like integration testing, this is primarily driven by the system's use cases and their scenarios. However, in many cases, integration testing requires modifications to the system. So, the focus of requirements testing is to ensure that the modi- fications made did not cause additional errors.

Usability testing is essentially a combination of the user interface and use-case testing that takes place during integration testing.

User interface and use-case testing focused on whether the user interface works and whether the use case was supported, respectively, usability testing focuses on how well the user interface supports the use cases. That is, how efficient and effective the user interface is. In many cases, this could include formal usability testing.

Documentation testing Given that documentation is basically a system in itself, documentation testing should involve both unit and integration testing. In this case, the unit is a documentation entry, and the user interface is either the paper or help screen. From an integration testing perspective, the focus is on whether the documentation works or not. And, like system testing of the soft- ware, the focus of system testing of the documentation is how well the documentation works. The reason that documentation is not typically tested in parallel with the system, i.e., when the classes, use cases, and user interface are tested, is to minimize the amount of documentation testing required. Even though the documentation should be developed in parallel with the software, until the software is tested, it is unclear exactly what to test in the documentation. As the software "passes" its tests, the documentation that goes along with the software can then be finalized and tested.

Performance testing focuses on trying to break the system with regard to the amount of work the system can handle. These types of tests typically fall into two categories:

1. *stress tests*: The purpose of stress tests, also known as load tests, is to ensure that the system can handle a certain number of simultaneous requests. For example, if the system is supposed to be able to handle 10,000 simultaneous requests, a stress test would attempt to push the system into handling more than that. If the performance of the test is insufficient, various software and database optimizations can be investigated. In other cases, additional hardware could be required.
2. *volume tests*: The purpose of volume tests is to push the implementation so that it may break when there is a large amount of data required to answer a user request. Again,

if it is discovered that the system fails this type of test, then database and software optimizations and additional hardware could be required. For exam- ple, sometimes it is more efficient to create a set of temporary tables by "selecting" the data from the actual tables before "joining" the tables together. By performing the "selects" first, the "join" works on less data. In this case, it could both speed up and lessen the amount of temporary storage required to handle the request. In other cases, denormalization of the data and storing the data at multiple locations could be called for. You typically do not want the user to make a request for a report and have to wait "too long" for the report to be processed. In some cases, giving up some functionality to improve performance can be crucial to the success of the system. So, the results of performance testing can make or break a system.

Security testing Security testing involves three primary areas:

1. authentication,
2. authorization, and
3. virus control.

Authentication testing deals with ensuring that the logged in user is who he or she claims to be. Typically, this has been addressed with user IDs and passwords and through the use of encryption techniques. Today, in addition to these approaches, various biometric identifiers have been used, e.g., retinal scans and fingerprints.

Authorization testing deals with ensuring that the logged in user actually has the authority to use the system(s) being accessed. Authorization has been controlled through the use of roles, access control lists, and capability lists. Security roles are the same as actor roles in a use-case model. Depending on the role being played by a user, different capabilities are made available to the user in the form of a capability list. However, in this case, a role can be specified down to the individual user level and not be limited to a group of users. Also, an access control list can be associated with each use case and with each class. In this case, an access control list specifies which roles have access to the resource (use case or class).

Virus controls Given that many system break-ins are a function of viruses, virus controls also need to be enforced. Anytime a file is received or sent by a user, the files should be scanned for potential viruses. This includes e-mail attachments, Web downloads, and the insertion of flash drives on desktop computers as well on all forms of “client” machines that can be attached to the system. Obviously, security requirements will impact the performance of the system. Therefore, trade-offs between these two sets of requirements may be necessary.