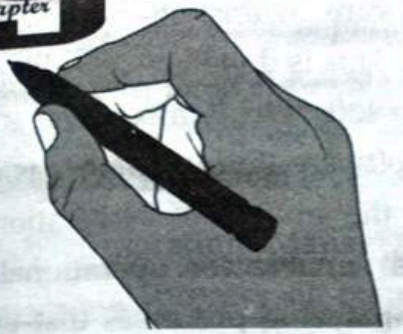
Chapter 9



SOFTWARE TESTING

CHAPTER OUTLINE

After studying this chapter, the reader will be able to understand the

- Software Verification and Validation
- Software Inspection
- Software Testing
- Software Testing Process
- Testing Methods
- Functional Testing
- Quality Management
- Software Measurement and Metric

SOFTWARE VERIFICATION AND VALIDATION

Software verification and validation activities check the software against its specifications. Every project must verify and validate the software it produces. This is done by: · checking that each software item meets specified requirements, checking each software item before it is used as an input to another activity, ensuring that checks on each software item are done, as far as possible, by someone other than the author and ensuring that the amount of verification and validation effort is adequate to show each software item is suitable for operational use. Verification and validation is the name given to checking an analyzing processes that ensure that software confirms to its specification and meets the need of customer who are paying for that software. It is the whole lifecycle process that starts with requirement review and continues through design review and code inspection to product testing. V & V are independent procedure that is used together for checking that a product service or system meets requirement specifications and that it fulfills its intended purpose. Whatever the size of project, software verification and validation greatly affects software quality. People are not infallible, and software that has not been verified has little chance of working.

Verification

Verification refers to the set of activities that ensure that software correctly implements specific functions. It ensures that the product has been built according to the requirement and design specification. It makes sure that the product is designed to deliver all functionality to the customer. Verification is done at the development process and includes reviews and meeting, inspection etc. to evaluate documents, plan code, requirements and specifications. Suppose you are constructing a table then verification is about checking all the parts of table, whether all legs are of correct size or not. It one leg is not of the correct size it will imbalance the end product. Verification is the demonstration of consistency, completeness and correctness of the software at each stage and between each stage of the development lifecycle. It is expressed as: Are we building the product right?

Advantages

- Verification helps in lowering down the count of the defect in the later stages of development.
- Verifying the product at the starting phase of development will help in understanding the product in better way.
- It reduces the chance of failure in software product.
- It helps in building the product as per the customer specification and needs.

Validation

Validation refers to the set of activities that ensure that software that has been built is traceable to the customer requirements. It is intended to show that the software does what the user requires. Validation concerns with determining if the system complies with the requirements and performs functions for which it is intended and meets the organizations goals and user needs. It is done at the end of the development process and takes place after verifications are completed. It answers the question like: Are we building the right product?

Advantages

- It some defects are missed in verification, then during validation defect can be caught as failures.
- If specification is misunderstood and development had happened then during validation process while executing that functionality the difference between the actual result and expected result can be understood.
- Validation helps in building the right product as per the customer's requirement and helps in satisfying their needs.
- Validation is basically done by the testers during the testing.

Difference between Verification and Validation

Verification	Validation
It is the process of evaluating product of development phase to find out whether they meet the requirement and design specification.	It is the process of evaluating software at the end to determine whether it meets the customer expectations and requirements.
Objective is to make sure that product is as per specification	Objective is to make sure that product actually meet up the user's requirements.
Activities involved are: review, meetings and inspections	Activities are testing like: black box testing , white box testing etc.
Verification is carried out by QA team	Validation is carried out by testing team.
Execution of code is not comes under verification	Execution of code comes under validation
Cost of error caught is less than validation	Cost of error caught is more than verification.
It is basically manual checking of documents like: requirement specification, design specification, test cases etc	It is basically checking of developed software by executing source code.
Are we building the system right?	Are we building the right system?

SOFTWARE INSPECTION

Software inspection is a control technique for ensuring that the documentation produced during a given phase remains consistent with the documentation of the previous phases and respects reestablished rules and standards. It is a technique to analyze and check system representation such as requirement document, design diagrams and program source code with executing it. It can be used as verification technique techniques before software is implemented. Inspection have proved to be an effective techniques of error detection, error can be found more cheaply through inspection than by extensive program testing. The aim of inspection is to locate faults and process should be driven by fault check list.

Inspection key Points

- It is most formal review type
- It is lead by trained moderator
- Documents are prepared and checked thoroughly by the reviewer.
- A separate preparation is carried out during which product is examined and defects are found
- It helps the author to improve the quality of the documents under inspection
- It removes the defects efficiently and as early as possible
- It improves product quality
- It creates common understanding by exchanging information
- It learn from defects found and prevent the occurrence of similar defects.

Inspection Roles

During an inspection, following roles are used

- **Author:** An author is the person who created the work product being inspected. Author is responsible for producing the program or document, fixing defects discovered during the inspection process.
- **Moderator:** Moderator is the leader of the inspection plans the inspection and coordinates it. Moderator reports process results to the chief moderator.
- **Chief Moderator:** Responsible for inspection process improvement, checklist updating, standards developments etc.
- **Reader:** Reader is the person reading through the documents one item at a time presents the code or documents at inspection meeting.

- **Recorder/Scribe:** The person that documents the defects that are found during the inspection is recorder.
- **Inspector:** Inspector is the person that examines the work product to identify possible defects, finds errors, omissions and inconsistencies in the code and documents.

Inspection Process

Inspections are a formal process used to identify and correct errors in a completed deliverable, before the deliverable is used as input to a subsequent deliverable. For example, after inspection, the Requirements Definition is released for reference by the Functional Design Specification. The focus of the inspection process is on finding defects, rather than solutions, which can divert the inspection meeting time. The inspection process is conducted by dividing an entire deliverable, such as the Requirements Definition, into manageable pieces that can be optimally inspected in a meeting. The stages in inspection process are: planning, overview, meeting, individual preparation, inspection meeting, rework and follow-up.

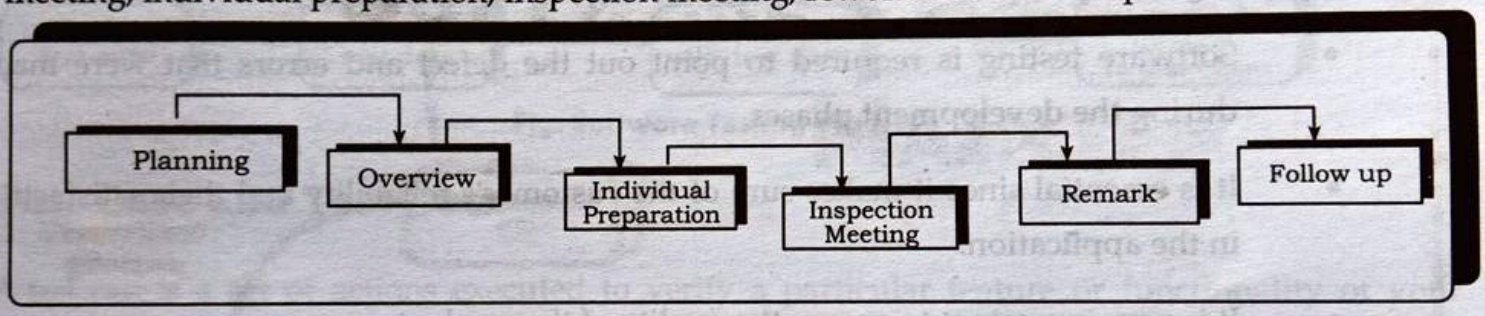


Fig: Software Inspection Process

1. **Planning:** The inspection is planned by the moderator which involves selecting an inspection team, organizing a meeting room and ensuring that the material to be inspected and its specifications are complete.
2. **Overview:** In this step, the software and documents to be inspected are presented to the inspection team where author of the code describes the background of the work product.
3. **Individual Preparation:** In this stage, each inspection team member studies the specification and the program and looks for the defects and identifies possible defects.
4. **Inspection Meeting:** During this meeting, the readers read through the work product part by part and inspectors point out defects for every part. Inspection meeting should be fairly short no more than two hours and should be focused on defect detection.
5. **Rework:** In this stage, the author makes changes to work product according to the action plan from the inspection meeting.
6. **Follow up:** In this step, the changes made by the author are checked to make sure that everything correct. The moderator should decide whether reinspection of the code and document is required. If not the software is then approved by moderator for release.

SOFTWARE TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. It is the process of executing software with the intent of finding software bugs and to ensure that it satisfies the specified requirements. Test is conducted for executing a system in order to identify any gaps, error or missing requirements in contrary to the actual desire or requirements.

Software is tested for validating and verifying software to ensure the business and technical requirements that guided its design and development and works as expected. It also provide an objective, independent view of software to allow the business to appreciate and understand the risk of software implementation.

Software Testing Necessicity

Software testing is very important because of following reasons.

- Software testing is required to point out the defect and errors that were made during the development phases.
- It is essential since it makes sure of the customer's reliability and their satisfaction in the application.
- It is very important to ensure the quality of the product.
- Testing is required for an effective performance of software product.
- Testing is necessary in order to provide the facilities to the customer like the delivery of high quality product or software application which requires lower maintenance cost.

Software Testing Objectives

The major objectives of software testing are as follows:

- Finding defects which may get created by the programmer while developing the software.
- Gaining confidence in and providing information about the level of quality.
- To make sure that the end result meets the business and user requirement.
- To gain the confidence of the customers by providing them a quality product.

SOFTWARE TESTING PROCESS

Software testing process starts with preparation of test cases. Test cases are specification of the inputs to the test and the expected output from the system together with what is being tested. From the design of test cases we get test case list. After that we prepare the test data. Test data are the inputs which have been devised to test the system. Test data can be generated automatically or manually. From this step we get test data set. Then we run the program with test data for all test cases and generate the result. Finally we compare the test result to test cases and generate the test report.

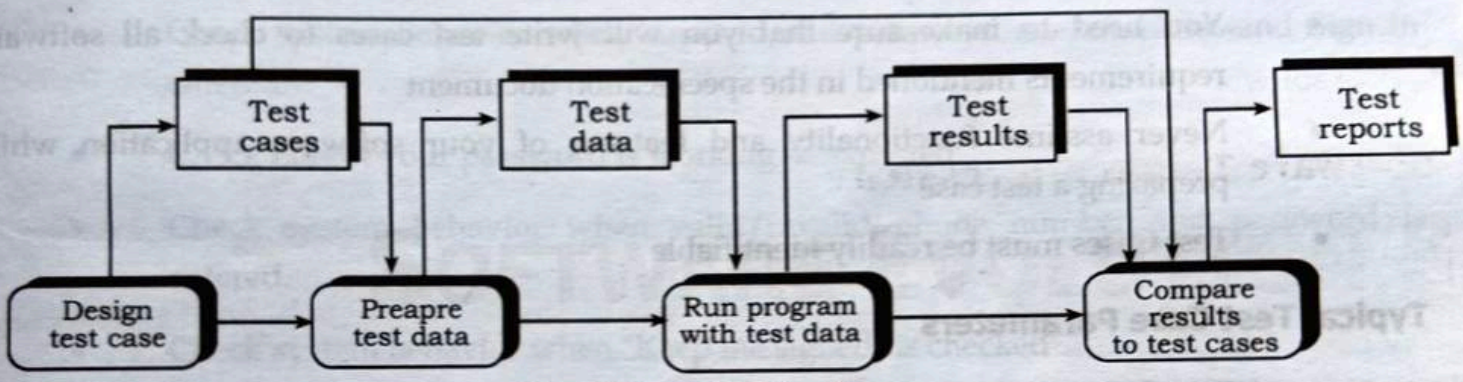


Fig: Software Testing Process

Test Case and Test Data

A **test case** is a set of actions executed to verify a particular feature or functionality of your software application. It is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test Case acts as the starting point for the test execution, and after applying a set of input values; the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

A Test Scenario is defined as any functionality that can be tested. It is a collective set of test cases which helps the testing team to determine the positive and negative characteristics of the project. Test Scenario gives a high-level idea of what we need to test.

Importance of test cases

- Test cases help to verify conformance to applicable standards, guidelines and customer requirements
- Helps you to validate expectations and customer requirements
- Increased control, logic, and data flow coverage

- You can simulate 'real' end user scenarios
- Exposes errors or defects
- When test cases are written for test execution, the test engineer's work will be organized better and simplified

Practices of Creating Test cases

- Test Cases should be transparent and straightforward
- Create Test Case by keeping the end user in the mind
- Avoid test case repetition
- You need to make sure that you will write test cases to check all software requirements mentioned in the specification document
- Never assume functionality and features of your software application while preparing a test case
- Test Cases must be readily identifiable

Typical Test Case Parameters

- Test Case ID
- Test Scenario
- Test Case Description
- Test Steps
- Prerequisite
- Test Data
- Expected Result
- Test Parameters
- Actual Result
- Environment Information
- Comments

Test Data

Test Data in Software Testing is the input given to a software program during test execution. It represents data that affects or affected by software execution while testing. Test data is used for both positive testing to verify that functions produce expected results for given inputs and for negative testing to test software ability to handle unusual, exceptional or unexpected inputs. Poorly designed testing data may not test all possible test scenarios which will hamper the quality of the software.

Example:

Test cases for the Test Scenario: ***Check the Login Functionality*** would be

- Check system behavior when valid email id and password is entered.
- Check system behavior when invalid email id and valid password is entered.
- Check system behavior when valid email id and invalid password is entered.
- Check system behavior when invalid email id and invalid password is entered.
- Check system behavior when email id and password are left blank and Sign in entered.
- Check Forgot your password is working as expected
- Check system behavior when valid/invalid phone number and password is entered.
- Check system behavior when "Keep me signed" is checked

Test Scenario ID	Login-1	Test Case ID	Login-1A
Test Case Description	Login-Positive test case	Test Priority	High
Pre-Requisite	A valid user account	Post-Requisite	NA

Test Execution Steps:

S. N.	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Results	Test Comments
1.	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	Launch successful
2.	Enter correct Email & password and hit login button	email ID: test@xyz.com Password *****	Login success	Login success	IE-11	Pass	Login successful

Test Scenario ID	Login-1	Test Case ID	Login-1B
Test Case Description	Login-Negative test case	Test Priority	High
Pre-Requisite	NA	Post-Requisite	NA

Test Execution Steps:							
S. N.	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Results	Test Comments
1.	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	Launch successful
2.	Enter invalid Email & password and hit login button	email ID: invalid@xyz.com Password *****	The Email address or phone number that you've entered doesn't match any account sign up for an account.	The Email address or phone number that you've entered doesn't match any account sign up for an account.	IE-11	Pass	Invalid Login attempt stopped
3.	Enter invalid Email & password and hit login button	Email ID: valid@xyz.com Password *****	The password that you've entered is incorrect. Forgotten password?	The password that you've entered is incorrect. Forgotten password?	IE-11	Pass	Invalid Login attempt stopped

Software Testing Levels

Testing levels are basically to identify missing areas and prevent overlap and repetition between development life cycle processes. In life cycle process, there are different phases and each phase goes through testing. Hence there are various levels of testing which are as follows:

1. **Unit Testing**

   It is basically done by the developers to make sure that their code is working fine and meet the user specifications. They test their piece of code which they have written like classes, functions, interfaces and procedures.

2. **Component Testing**

   The component testing is concerned with testing the functioning of clearly identifiable components. It is also called as module testing. The basic difference between unit testing and component testing is, in unit testing the developers test their piece of code but in component testing whole component is tested. For e.g. in a student record application, there are two modules one which will save records and other module is to upload the result of students both the modules are developed separately and when they are tested one by one then we call this as a component or module testing. Software components are often composite components that are made up of several interacting objects. Testing components should therefore emphasize on showing that the component interface behaves according to its specification. There are different types of interface between program components and consequently different types of interface error that can occur.

a. **Parameter Interface:** these are interfaces that are used to pass data or references form one function to another.

b. **Procedural Interfaces:** These are interfaces in which one component encapsulates a set of procedures that can be called by another component.

c. **Shared Memory Interface:** These are interfaces in which block of memory is shared between components of software. In shared memory data is placed by one component and retrieved from there by other subsystem.

d. **Message Passing Interface:** These are interfaces in which one component request a service from another component by passing message to it.

Different interface errors that may occur in component are as follows:

a. **Interface Misuse:** This type of error occurs when a calling component calls some other component and makes an error in the use of its interface. This type of error occurs with parameter interfaces.

b. **Interface Misunderstanding:** This type of error occur when a calling component misunderstands the specification of the interface of the called component and makes assumptions about its behavior.

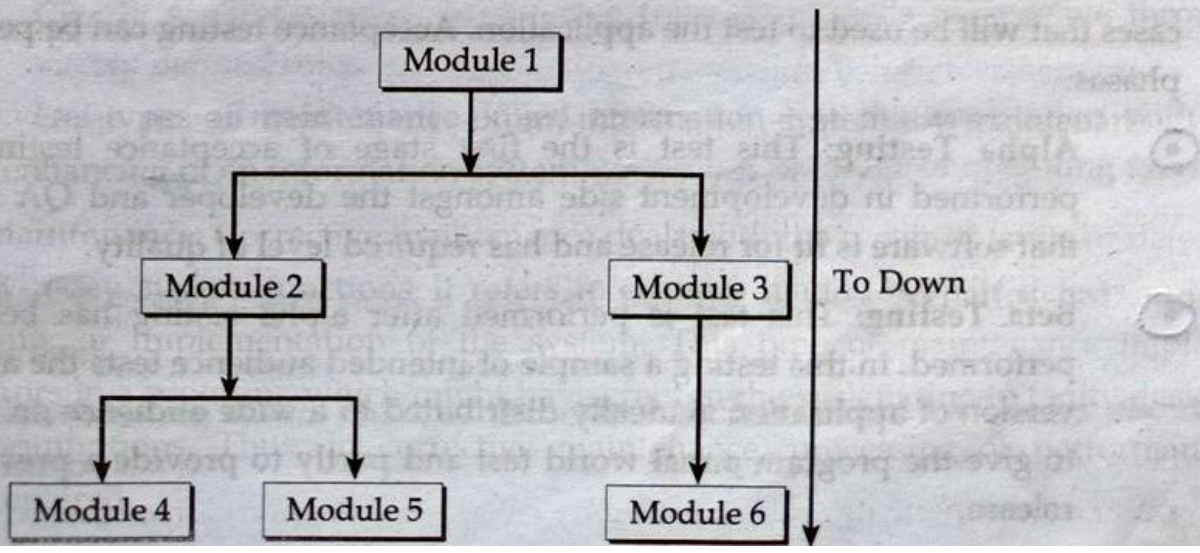
c. **Timing Error:** This type of error occurs in real time systems which use shared memory or message passing interface when producer of data and the consumer of data may operate at different speeds.

3. Integration Testing

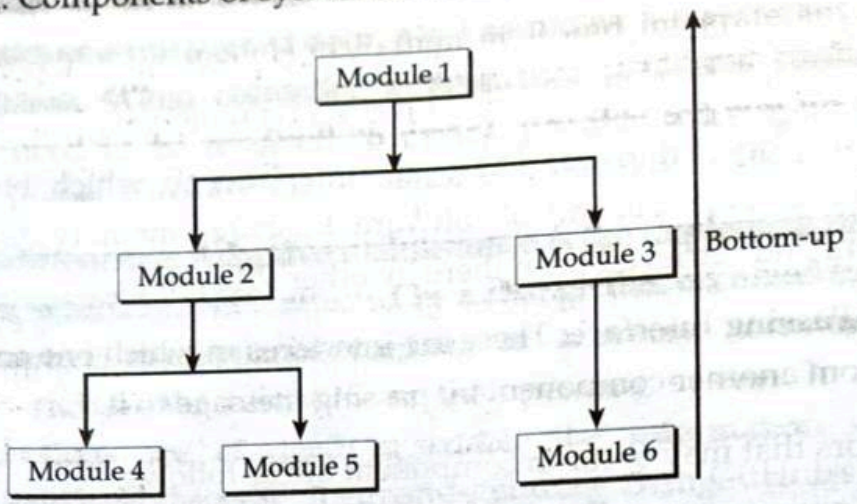
Integration testing is done when two modules are integrated, in order to test the behavior and functionality of both the modules after integration. Defects in the component that have been missed during component testing are discovered during integration testing.

Integration testing follows two approaches known as top down and bottom up approach.

a. **Top-down integration:** Testing takes place from top to bottom, following the control flow or architectural structure eg: starting from GUI or main mean. Components or systems are substituted by stubs.



b. **Bottom-up integration:** Testing takes place from the bottom of the control flow upward. Components of system are substituted by drivers.



4. **System Testing**

In system testing the behavior of whole system/product is tested as defined by the scope of the developed project or product. It may include tests based on risks, requirement specifications, business process, other high level descriptions of system behavior, interactions with operating system etc. System testing is most of the final test top verify that the system to be delivered meets the specification and purpose. It should investigate both functional and non-functional requirements. System testing is carried out by specialist tester or independent testing team with no involvement from designers and programmers. System testing should focus on testing interactions between the components and objects that make up a system as well as test the reusable components or system to check that they work as expected when they are integrated with new components.

5. **Acceptance Testing**

After the system test has corrected all or most defects, the system will be delivered to the user or customer for acceptance testing. The goal of acceptance testing is to establish confidence in the system, it is most often focused on a validation types testing. This is arguably most important type of testing as it is conducted by the quality assurance team who will check whether the application meets the intended specifications and satisfies the client requirements. The QA team will have a set of prewritten scenarios and test cases that will be used to test the application. Acceptance testing can be performed in two phases:

- **Alpha Testing:** This test is the first stage of acceptance testing and will be performed in development side amongst the developer and QA team to ensure that software is fit for release and has required level of quality.
- **Beta Testing:** This test is performed after alpha testing has been successfully performed. In this testing a sample of intended audience tests the application. Beta version of application is ideally distributed to a wide audience on the web, partly to give the program a real world test and partly to provide a preview of the next release.

TESTING METHODS

Testing can be performed without having any knowledge of the interior workings of the application as well as by having knowledge of implementation detail of software system. There are two testing methods which are black box testing and white box testing.

Black Box Testing

Black box testing is the technique of software testing in which without having knowledge of the interior working, application is tested to check the relationship between its input and output. In this testing, knowing the specified function that a product has been design to perform, test can be conducted that demonstrate each function is fully operational. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a testing will interact with the system's user interface by providing inputs and examining output without knowing how and where the inputs are worked upon. In black box testing, system is like black box whose behavior can only be determined by studying its input and related outputs.

Black box testing is also called functional or behavioral testing focuses on functional requirement of software i.e. tester is only concerned with functionality not the implementation of software. This testing is especially applicable to perform system testing. Black box testing attempts to find errors in the following categories:

- Incorrect or missing functions.
- Interface error.
- Error in data structure.
- Behavior or performance
- Initialization or termination error.

Advantages

- Well suited and efficient for large code segments.
- Code access is not required.
  - Clearly separates user's perspective from developer's perspective through visibly defined roles.
  - Large no. of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating system.

Disadvantages

- Limited coverage since only a selected number of test scenarios are actually performed.
- Inefficient testing due to fact that the tester only has limited knowledge about an application.

- Blind Coverage, since the tester cannot target specific code segments or error prone areas.
- The test cases are difficult to design.

White Box Testing

White box testing is an approach to testing where the tests are derived from knowledge of software's structure and implementation. It is the detailed investigation of internal logic and structure of the software system. In this testing, knowing the internal working of the product, test case can be conducted to ensure that, internal operations are performed according to specifications and all internal components have been adequately exercised. In order to perform white box testing in an application, the tester needs to process knowledge of internal workings of the code, the tester needs to have look inside the sources code and find out which unit chunk of the code is behaving in appropriately. Using white box testing, the software can derive test cases that:

- Guarantee that all independent paths with a module have been exercised at least once.
- Exercise all logical decisions on their true and false value.
- Execute all loops at their boundaries.

This method is usually applied to relatively small program unit, subroutine and operations such as unit testing, component testing and integration testing. It is also known as clear box testing, glass box testing or structural testing.

Advantages

- As the tester has knowledge of source code, it becomes very easy to find out which type of data can help in testing in optimizing the code.
- It helps in optimizing the code.
- Extra lines of codes can be removed which can bring in hidden defects.
- Due to tester's knowledge about the code, maximum coverage is attainted during test scenario writing.

Disadvantages

- Due to fact that a skilled tester is needed to perform white box testing, the costs are increased.
- Sometimes it is impossible to look into every nook and corners to find out hidden errors that may create problems as many paths will go untested.
- It is difficult to maintain as the use of specialized tools like code analyzers and debugging tools are required.

Difference between Black box Testing and White box Testing

Black box Testing	White box Testing
1. The internal workings of an applications are not required to be known	1. Tester has full knowledge of the internal working of the application.
2. Also known as closed –box, data driven or functional testing.	2. Also known as clear box, structural or code based testing.
3. performed by end users and also by testers and developers	3. Normally done by testers and developers.
4. Testing is based on external expectations; internal behavior of application is unknown.	4. Internal workings are fully known and tester can design test data accordingly.
5. This is least time consuming and exhaustive.	5. The most exhaustive and time consuming.
6. Not suited to algorithm testing	6. Suited for algorithm testing.
7. This can only be done by trial and error method	7. Data domain and internal boundaries can be better tested.
8. This type of testing always focus on what is performing or carried out	8. This type of testing always focuses on how it is performing.
9. Mainly applicable to higher levels of testing such as system testing and acceptance or release testing.	9. Mainly applicable to lower levels of testing such as unit testing, integration testing etc.
10. It is performed by testing team	10. It is performed by developer themselves.
11. Basis of test case is requirement specification.	11. Basis of test case is detail design of the system.

FUNCTIONAL TESTING

This is the type of black box testing that is based on the specifications of the software that is being testing. The applications is tested by providing input and then the results are examined that need to conform to functionality it was intended for. Functional testing of software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. There are five steps that are involved when testing an application for functionality.

- The determination of functionality that the intended application is meant to perform.
- The creation of test data based on specifications of the application.
- The output based on test data and the specification of the application.
- The writing of test scenarios and executing of test cases.
- The comparison of actual and expected results based on executed test cases.

Non-functional Testing

Non-functional testing is based upon the testing of the application from its non-functional attributes such as performance, security, user interface etc.

Performance Testing

Performance test have to be designed to ensure that the system can process its intended load. This actually involves planning a series of tests where the load is steadily increased until system performance becomes unacceptable. Performance testing is concerned with both demonstrating that the system meets its requirement and discovering problems and defects in the system. There are different causes which contribute in lowering the performance of software.

- Network delay
- Client side processing
- Database transaction processing
- Load balance between servicers
- Data rendering

Performance testing is considered as one of the important and mandatory testing type in terms of aspects such as: Speed, capacity, stability, scalability. Performance testing can be qualitative or quantitative and can be divided into two sub-types such as load testing and stress testing.

Path Testing

Path testing is a structural testing strategy whose objective is to exercise every independent execution path through a component. The objective of path testing is to ensure that the set of test cases is such that each path through the programs is executed at least once.

All conditional statements are tested for both true and false value and cases. The starting point for path testing is a program how graph. This is a skeletal model of all paths through the program. A how graph consist of nodes representing decision and edges showing how of control.

QUALITY MANAGEMENT

Software quality management is concerned with ensuring that software has a low number of defects and that it reaches the required standard of maintainability, reliability, and portability and so on. It concerned with ensuring that the required level of quality is achieved in a software product. Quality management involves defining appropriate quality standards and procedures and ensuring that these are followed and should aim to develop a quality culture where quality is seen as everyone's responsibility. Quality management is particularly important for large and

complex system in which quality documentation is prepared which is record of progress and supports continuity of development as the development changes. For smaller system, quality management needs less documentation and should focus on establishing a quality culture and ensuring that all team members have a positive approach to software quality. Software quality management for large systems can be structure into three main activities.

Quality Assurance

Software quality assurance is the establishment of a framework of organizational procedures and standards that lead to high quality software. It is the process of defining how software quality can be achieved and how the development organization known that software has required level quality. Quality assurance is primarily concerned with defining or selecting standards that should be applied to software development process or product. Two types of standards that may be used as part of QA.

- **Product Standards:** These stands apply to the software product being development they include document standards, such as structure of requirement document, coding standard that define how a programming language is used etc.
- **Process Standard:** These standards define the process that should be followed during software development they may include definition of specification, design and validation process and description of documents that should be written in the course of these process.

Quality Planning

Quality planning is the process of selecting appropriate procedure and standards from the framework and adapted for a specific software project and modifies these as required.

It is the process of developing a quality plan for a project. It includes the desired software qualities and describes how these are to be accessed. So, quality planning defined what high quality software actually means.

The quality planning should select those organizational standards that are appropriate to a particular product and development process. It varies on type of and size of system that is being developed. The quality plan defines the most important quality attributes for the software and includes a definition of the quality assessment process. Some of the software quality attributes that can be considered during the quality planning process are listed below: Safety, understandability, Portability, Security, Testability, Usability, Reliability, Adaptability, Reusability, Resilience, Modularity, Efficiency, Robustness, Complexity, Learnability, Maintainability

The quality plan should include:

- **Product intro:** It includes description of product, its intended market and quality expectations for the product.

- **Project plan:** It includes critical release dates and responsibilities for product along with plan for distribution and product servicing.
- **Process description:** It includes development and service process that should be used for product development and management.
- **Quality Goals:** It includes the quality goals and plans for the product including and identification and justification of critical product quality attributes.
- Risk and Risk Management: It includes key risk that might affect product quality and action to address these risks.

Quality Control

Quality control provides monitoring the software development process to ensure that quality assurance procedures and standards are being followed. The deliverables from the software development process are checked against the defined project standards in the quality control process. The quality of software project deliverables can be checked by regular quality reviews and/or automated software assessment. Quality reviews are performed by a group of people. They review the software and software process in order to check that the project standards have been followed and that software and documents conform to these standards. in review process different types of review are design or program inspections, progress review and quality review.

Automated software assessment processes the software by a program that compares it to the standards applied to the development project. This assessment involves a quantitative measurement of some software attributes.

Software Measurement and Metric

A measurement is a manifestation of the size, quantity, amount or dimension of a particular attributes of a product or process. Software measurement is a titrate impute of a characteristic of a software product or the software process. Software measurement can be used to gather quantitative data about software and the software process.

Software is measured to:

- Create the quality of the current product or process.
- Anticipate future qualities of the product or process.
- Enhance the quality of a product or process.
- Regulate the state of the project in relation to budget and schedule.

Software Measurement Process

Software measurement process is a part of quality control process in which each components of the system is analyzed separately and different values of their metric are compared with each other as well as historical data that were collected from previous projects. The key stages in measurement process are:

a. **Choose measurement to be made:** In this stage, the measurement question is formulated to which measurement required to answer.

b. **Select component to be assessed:** In this stage, the critical components of system are selected for finding the metric values.

c. **Measure component characteristics:** In this stage, the selected components are measured and metric values are computed using automated tools.

d. **Identify anomalous measurements:** After finding the metric values for all the selected components, these should be compared to each other and to previous measurements which have been already recorded.

e. **Analyze anomalous components:** In this stage, after finding the anomalous components and their metric valuses it is decided whether quality is comprised in those component s or not.

Product Metrics

Software product metrics are measures of software products such as source code and design documents. These are measures of software development process and concerned with characteristics of the software itself. For example, the size of the software is a measure of the software product itself. Products metric are of two types:

Dynamic metrics are the metrics collected by measurement made of a program in execution. These are usually quite closely related to software quality attributes. It is relatively easy to measure the execution time required for particular tasks and to estimate the time required to start the system.

On the other hand, static matrices are collected by measurements made of the system representation such as design; documentation and source code etc. These have an indirect relationship with quality attributes. A large number of these matrices have been proposed to try to derive and validate the relationship between the complexity, understandability, and maintainability.

ISO9000

The International Standards Organization (ISO) is a worldwide federation of national standards bodies. The work of preparing international standards is carried out through ISO technical

committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. In liaison with International Standards Organization, international organizations (governmental and nongovernmental) also take part in the work. The emphasis for international standards originated in the European Community's plan to become a single market with international standards becoming effective at the end of 1992. At that time, organizations wishing to do business in the common market were required to meet those standards, which are now accepted worldwide. There are five related quality management standards in what is referred to as the ISO 9000 series. Each of the five standards (numbered 9000 to 9004) addresses a different topical area. For example, ISO 9000 affects the development and maintenance of software. In addition, the International Standards Organization 9000 standard provides some basic definitions and concepts. It summarizes the other standards in the series and explains how to select and use them. The International Standards Organization 9001, 9002, and 9003 standards ensure external quality in contractual situations. ISO 9004 contains guidance on the technical, administrative, and human factors affecting the quality of products and services.

The five standards in the ISO 9000 series are:

- **ISO 9000-1:** Quality management and quality assurance standards—Part 1: guidelines for selection and use.
- **ISO 9001:** Quality systems—model for quality assurance in design, development, production, installation, and servicing.
- **ISO 9002:** Quality systems—model for quality assurance in production, installation, and servicing.
- **ISO 9003:** Quality systems—model for quality assurance in final inspection and test.
- **ISO 9004-1:** Quality management and quality system elements—Part 1: guidelines.

CMM (Capability Maturity Model)

It has long been accepted that continuous process improvement is based on many small evolutionary steps rather than larger revolutionary innovations. The Capability Maturity Model (CMM) provides a framework for organizing these evolutionary steps into five maturity levels that lay successive foundations for continuous process improvement. This methodology is at the heart of most management systems which are designed to improve the quality of the development and delivery of all products and services.

The Capability Maturity Model for Software provides software organizations with guidance on how to gain control of their process for developing and maintaining software and how to evolve toward a culture of software engineering excellence. The CMM was designed to guide software organizations in selecting process improvement strategies by determining current process maturity and identifying the few issues most critical to software quality and process improvement.

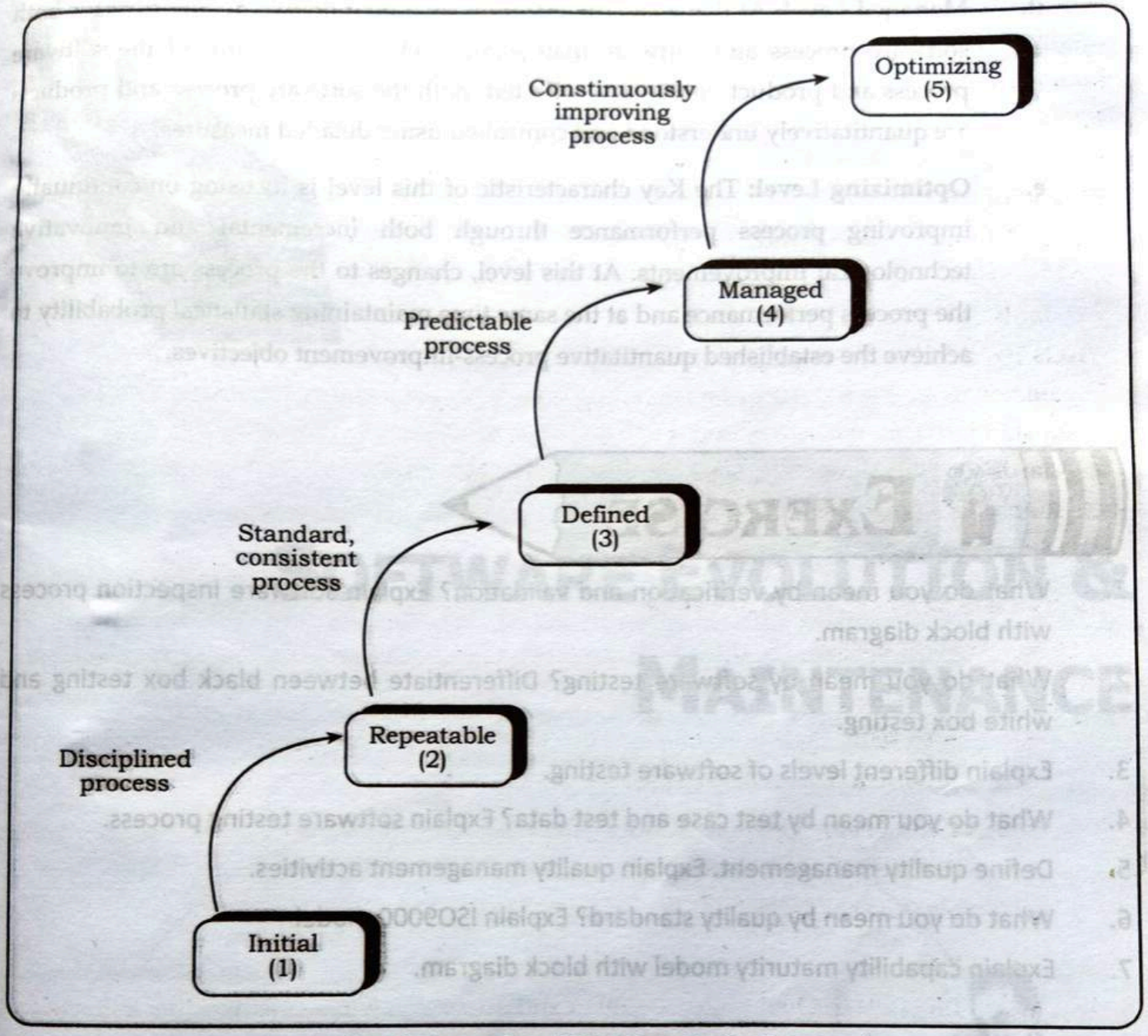


Fig: Software Capability Maturity Model

The following characterizations of the five maturity levels highlight the primary process changes made at each level.

a. **Initial Level:** The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

b. **Repeatable Level:** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

c. **Defined Level:** The software process for both management and engineering activities is documented, standardized, and integrated into an organization-wide software process. All projects use a documented and approved version of the organization's process for developing and maintaining software.

d. **Managed Level:** At this level, organization set a quantitative quality goal for both software process and software maintenance. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures.

e. **Optimizing Level:** The Key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements. At this level, changes to the process are to improve the process performance and at the same time maintaining statistical probability to achieve the established quantitative process-improvement objectives.

EXERCISE

1. What do you mean by verification and validation? Explain software inspection process with block diagram.
2. What do you mean by software testing? Differentiate between black box testing and white box testing.
3. Explain different levels of software testing.
4. What do you mean by test case and test data? Explain software testing process.
5. Define quality management. Explain quality management activities.
6. What do you mean by quality standard? Explain ISO9000 model
7. Explain capability maturity model with block diagram.

❑❑❑