SOFTWARE ENGINEERING

Unit 4: Software Testing

Shree Harsh Attri
(Assistant Professor)
Department: Bachelor of Computer Applications

JIMS Engineering Management Technical Campus
Greater Noida (U.P)

WHAT IS TESTING?

"Testing is the process of executing a program with the intent of finding errors."

Why Testing is needed ?

In terms of software engineering, software testing is an expensive activity, yet launching of software without testing may lead to cost potentially much higher than that of testing, specially in systems where human safety is involved.

In the software life cycle, ***the earlier the errors are discovered and removed, the lower is the cost of their removal***.

Who should Do the Testing ?

Testing requires the developers to find errors from their software.

It is difficult for software developer to point out errors from own creations.

Many organisations have made a distinction between development and testing phase by making different people responsible for each phase.

WHAT IS SOFTWARE TESTING?

jims Greater Noida

- It an activity to check whether the actual results match the expected results and to ensure that the software system is **Defect Free**.
- It involves execution of a *software component* or *system component* to evaluate one or more properties of interest.
- Software testing also *helps to identify errors, gaps or missing requirements in contrary to the actual requirements*. It can be either done manually or using automated tools.
- In other terms Software Testing can also be called as **Verification of Application Under Test (AUT).**
- The process of software testing aims not only at finding faults in the existing software but also at ***finding measures to improve the software in terms of efficiency, accuracy and usability***.
- It mainly ***aims at measuring specification, functionality and performance of a software program or application***.

WHY SOFTWARE TESTING IS IMPORTANT?

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause *monetary and human loss*, for examples:

- In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
- Nissan cars have to recall over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.
- Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point store served coffee for free as they unable to process the transaction.
- In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.
- China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocent live
- In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
- In April of 1999, a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history
- In may of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.

jims Greater Noida

Some Terminologies

- **Error, Mistake, Bug, Fault and Failure**

People make **errors**. A good synonym is **mistake**. This may be a syntax error or misunderstanding of specifications. Sometimes, there are logical errors.

When developers make mistakes while coding, we call these mistakes "**bugs**".

A **fault** is the representation of an error, where representation is the mode of expression, such as narrative text, data flow diagrams, ER diagrams, source code etc. Defect is a good synonym for fault.

A **failure** occurs when a fault executes. A particular fault may cause different failures, depending on how it has been exercised.

SOFTWARE TESTING

jims Greater Noida

➢ Test, Test Case and Test Suite

Test and **Test case** terms are used interchangeably. In practice, both are same and are treated as synonyms. Test case describes an input description and an expected output description.

Test Case ID	
Section-I (Before Execution)	Section-II (After Execution)
Purpose :	Execution History:
Pre condition: (If any)	Result:
Inputs:	If fails, any possible reason (Optional);
Expected Outputs:	Any other observation:
Post conditions:	Any suggestion:
Written by:	Run by:
Date:	Date:

Test Case Template

The set of test cases is called a **test suite**. Hence any combination of test cases may generate a test suite.

SOFTWARE TESTING

jims Greater Noida

What is Code Review?

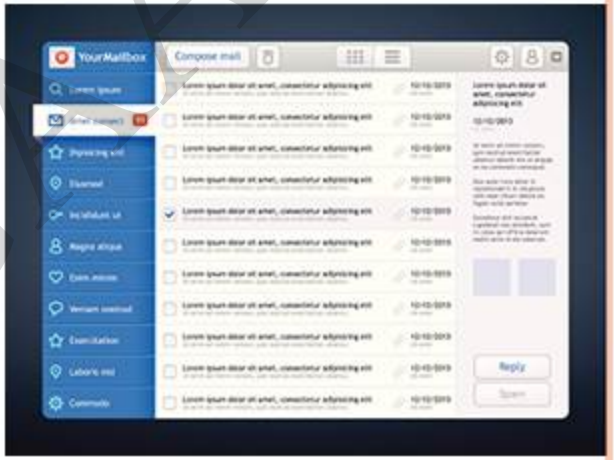
- Code Review is a systematic examination, which can find and remove the vulnerabilities in the code such as memory leaks and buffer overflows.
- ***Technical reviews*** are well documented and use a well-defined defect detection process that includes peers and technical experts.
- This kind of review is usually performed as a peer review without management participation.
- Reviewers prepare for the review meeting and prepare a review report with a list of findings.
- ***Technical reviews*** may be quite informal or very formal and can have a number of purposes but not limited to discussion, decision making, evaluation of alternatives, finding defects and solving technical problems.
- Code reviews, expose developers to new ideas and technologies, so that they can write better and better code.

SOFTWARE TESTING

jims Greater Noida

Common Code Review Approaches are:

Email Thread: As soon as a given piece of code is ready for review, the file is sent around to the appropriate colleagues via email for each of them to review as soon as their workflow permits. While this approach can certainly be more flexible and adaptive than more traditional techniques, such as getting five people together in a room for a code-inspection meeting.



Pair Programming: this approach for writing software puts developers side by side, working on the same code together and thereby checking each other's work as they go. It's a good way for ***senior developers to mentor junior colleagues***, and seems to bake code review directly into the programming process.

SOFTWARE TESTING

Common Code Review Approaches are:

Over-the-Shoulder: More comfortable for most developers than pair programming, the old over-the-shoulder technique is the easiest and most intuitive way to engage in peer code review. Once your code is ready, just find a qualified colleague to site down at your workstation (or go to theirs) and review your code for you, as you explain to them why you wrote it the way you did.

Tool-Assisted: more efficient way to review code is through software-based code review tools, some of which are browser-based or seamlessly integrate within a variety of standard IDE and SCM development frameworks. Software tools solve many of the limitations of the preceding approaches above, tracking colleagues' comments and proposed solutions to defects in a clear and coherent sequence.

SOFTWARE TESTING

jims Greater Noida

Software testing can be divided into two steps:

1. **Verification:** it refers to the set of tasks that ensure that software correctly implements a specific function.

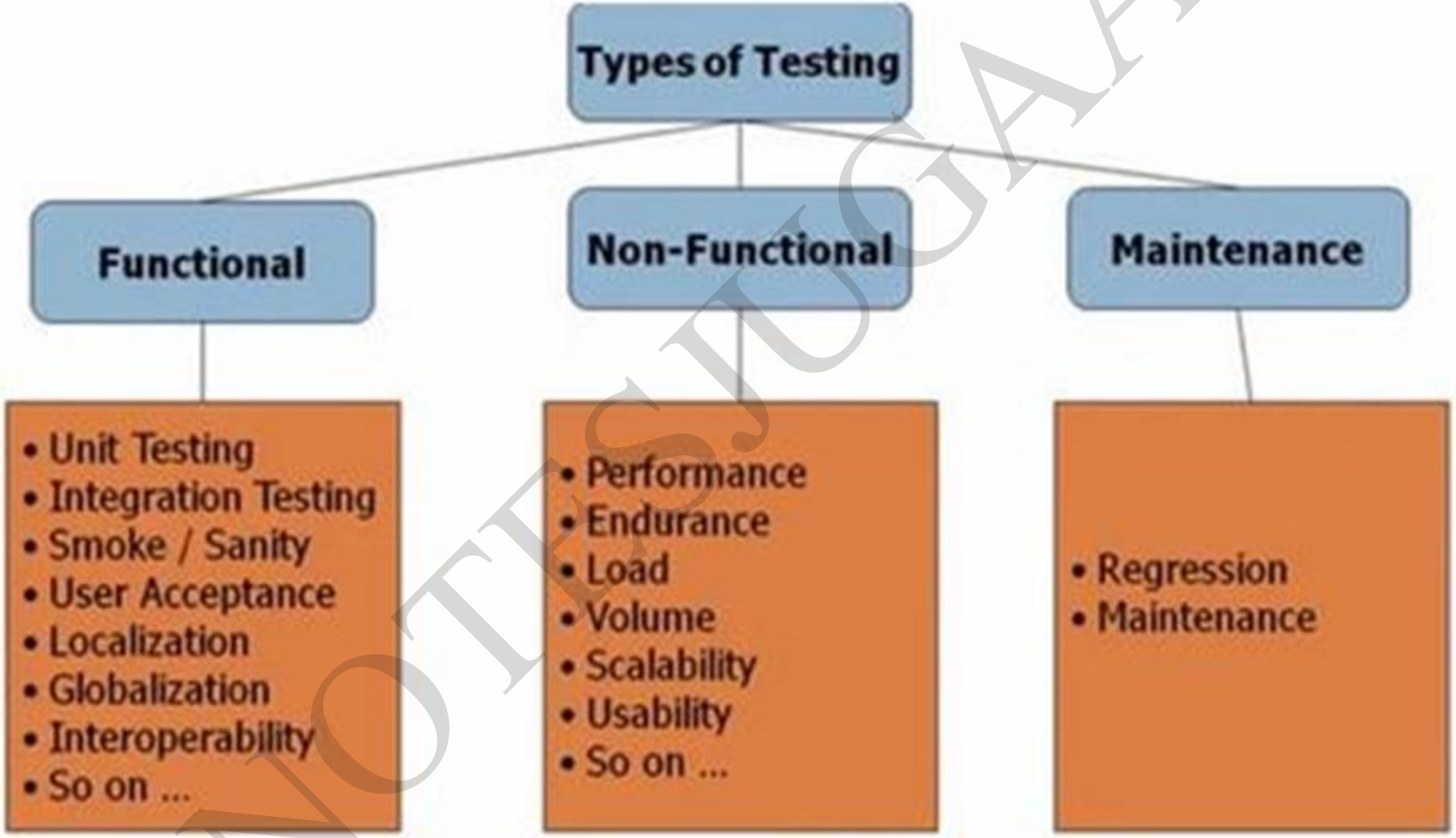
2. **Validation:** it refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

Verification : Are we building the product right?
Validation : Are we building the right product?

Testing = Verification + Validation

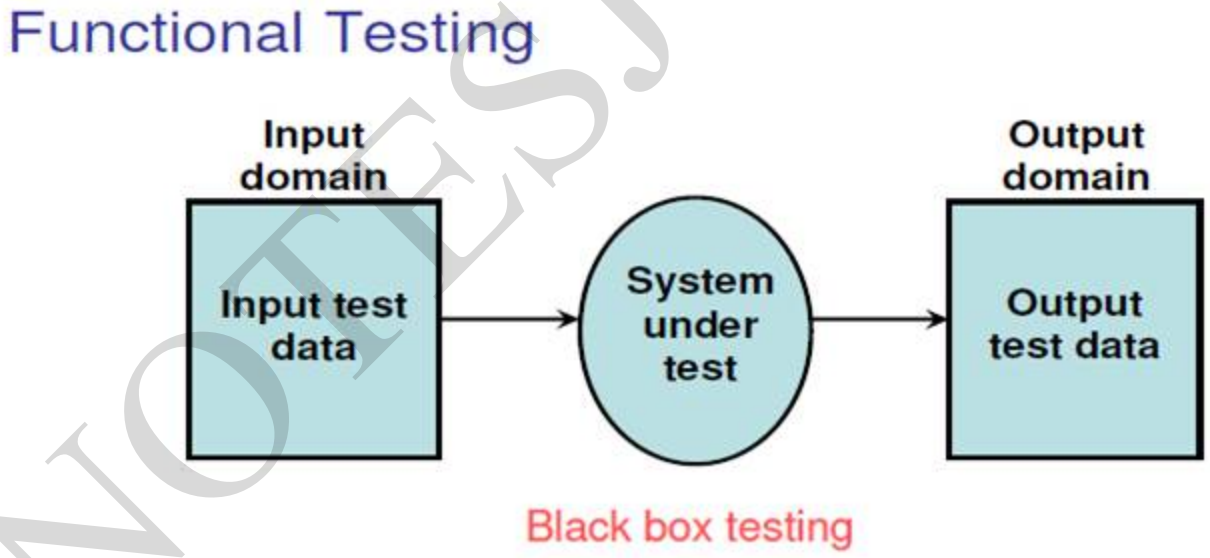
SOFTWARE TESTING

Types of Software Testing



Functional Testing

1. Functional tests are processes designed to confirm that all of the components of a piece of code or software operate correctly.
2. Functional testing focuses on testing the interface of the application to ensure that all user requirements for a properly working application are met.
3. This type of testing is not concerned with how processing occurs, but rather, with the results of processing.
4. It is also known as **Black Box Testing.**



Functional Testing

Functional testing involves the following steps:

a) Identify functions that the software is expected to perform.
b) Create input data based on the function's specifications.
c) Determine the output based on the function's specifications.
d) Execute the test case.
e) Compare the actual and expected outputs.

Advantages of Functional Testing

a) It acts as actual system usage.
b) It does not make any system structure assumptions.

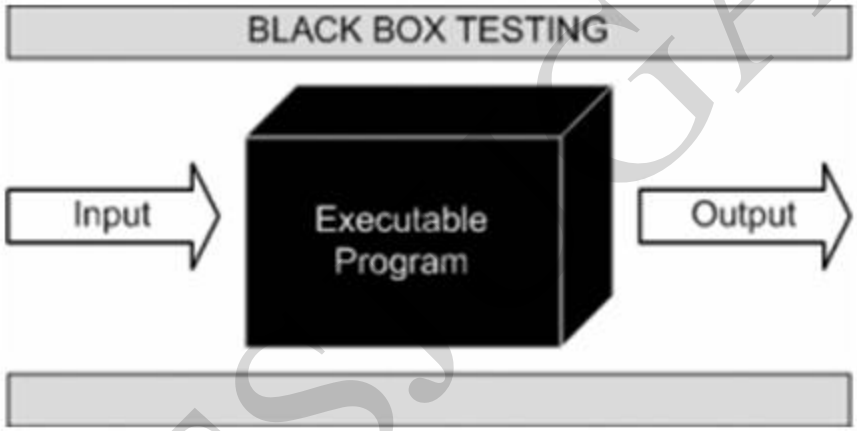
Disadvantages of Functional Testing

a) It has a potential of missing logical errors.
b) It has a high possibility of redundant testing.

SOFTWARE TESTING

Black Box Testing

It is also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester.



This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

Structural Testing: also called White Box Testing

It's a testing used to test the structure of coding of software. This type of testing requires knowledge of the code, so, it is mostly done by the developers. It is more concerned with how system does it rather than the functionality of the system.
The other names of structural testing includes glass box testing, clear box testing, open box testing, logic driven testing or path driven testing.

Structural Testing Techniques:

1. **Statement Coverage** - This technique is aimed at exercising all programming statements with minimal tests.

2. **Branch Coverage** - This technique is running a series of tests to ensure that all branches are tested at least once.

3. **Path Coverage** - This technique corresponds to testing all possible paths which means that each statement and branch are covered.

Types of Structural Testing

The structural testing is based on different types of approaches as follows:

1. **Control flow testing:**
   The basic model of the testing is the flow of control. The whole test is based on how the control is carried out throughout the program. This method requires detailed knowledge of all aspects of the software and the logic of the software. It tests out the whole code thoroughly.

2. **Data flow testing:**
   This implements the use of a control flow graph and checks the points where the codes can lead to an alteration in the data. In this way, the data is kept safe and unaltered throughout the execution of the program. Any alteration of the data can result in adverse consequences.

Types of Structural Testing

The structural testing is based on different types of approaches as follows:

3. Slice based testing:

It was originally created and developed for maintaining the software. The basic idea is to divide the whole program into small slices and then checking on to each slice carefully. This method is very useful for the maintenance of the software as well as debugging the software too.

4. Mutation testing:

This is the type of software testing that requires the development of new tests to be carried out on the software for its testing. The developers make small alterations to the already available software tests and create a mutant of the old software test. This is how the name mutation testing arises. The developer then carries out the mutation tests on the program as he wishes to.

Structural Testing:

Advantages of Structural Testing:

1. Forces test developer to reason carefully about implementation
2. Reveals errors in "hidden" code
3. Spots the Dead Code or other issues with respect to best programming practices.

Disadvantages of Structural Box Testing:

1. Expensive as one has to spend both time and money to perform white box testing.
2. Every possibility that few lines of code is missed accidentally.
3. Indepth knowledge about the programming language is necessary to perform white box testing.

Testing Activities

Fundamental Test processes consist of Five Activities:

1. **Planning:** Test Planning activity produces a test plan specific to a level of testing. These test levels specifies how the test strategy and project test plan apply to that level of testing and state any exceptions to them.
   The test plan must state the completion criteria to determine when this level of testing is complete.

Example of completion criteria are:

a) 100% requirement coverage;
b) all screens, dialogue boxes and error messages seen;
c) 100% of test cases have been run;
d) 100% of high severity faults fixed;

jims Greater Noida

Testing Activities

Fundamental Test processes consist of Five Activities:

2. **Specification:** The fundamental activity is to design the test cases using the techniques selected during planning. ***For each test case, specify its objective, the initial state of the software, the input sequence and the expected outcome.***

Specification can be considered as three separate tasks:

a) Identify test conditions
b) Design test cases – determine 'how' the identified test conditions are going to be exercised;
c) Build test cases – implementation of the test cases (scripts, data, etc.).

Testing Activities

Fundamental Test processes consist of Five Activities:

3. **Execution:** The purpose of this activity is to execute all of the test cases. This can be done either manually or with the use of a test execution automation tool.

   Test Monitoring and Control is the process of overseeing all the metrics necessary to ensure that the project is executed well, on schedule, and not out of budget.

   Monitoring: Monitoring is a process of **collecting**, **recording**, and **reporting** information about the project activity that the project manager and stakeholder needs to know.

   To Monitor, Test Manager does following activities

   a) Define the project goal, or project performance standard
   b) Observe the project performance, and compare between the actual and the planned performance expectations
   c) **Record** and **report** any detected problem which happens to the project

Testing Activities

Fundamental Test processes consist of Five Activities:

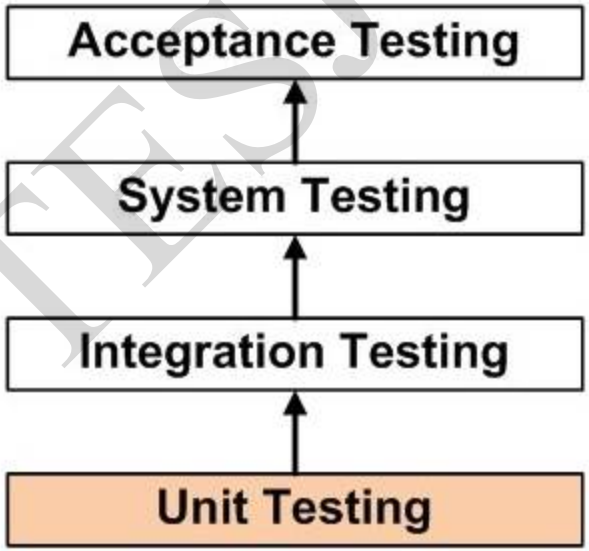
4. **Recording:** Test Recording activity is done in parallel with Test Execution. It is needed to record the versions of the software under test and the test specification being used. Then for each test case it should record that what is the actual outcome and what are the test coverage levels achieved for those measures specified as test completion criteria in the test plan.
   The actual outcome should be compared against the expected outcome and if any discrepancy found the that must be analysed in order to establish where the fault lies.

5. **Checking for Test Completion:** This activity has the purpose of checking the records against the completion criteria specified in the test plan. If these criteria are not met, it will be necessary to go back to the specification stage to specify more test cases to meet the completion criteria. There are many different types of coverage measure and different coverage measures apply to different levels of testing.

SOFTWARE TESTING

Unit Testing:

1. It is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected.
2. Unit Testing is done during the development (coding phase) of an application by the developers and it comes under white box testing.
3. A ***unit*** considered as an *individual function, method, procedure, module or object.*



Unit Testing Levels

Unit Testing:

Objectives:

1. To isolate a section of code.
2. To verify the correctness of code.
3. To test every function and procedure.
4. To fix bug early in development cycle and to save costs.
5. To help the developers to understand the code base and enable them to make changes quickly.
6. To help for code reuse.

Advantages:

1. Reduces Defects in the Newly developed features or reduces bugs when changing the existing functionality.
2. Reduces Cost of Testing as defects are captured in very early phase.
3. Improves design and allows better refactoring of code.
4. Unit Tests, when integrated with build gives the quality of the build as well.

Integration Testing:

1. It is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers.
2. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.
3. Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as **'I & T'** (Integration and Testing), **'String Testing'** and sometimes **'Thread Testing'**.

Integration Strategies / Approaches:

1. Big-Bang Integration
2. Incremental Approaches:
   a) Top Down Integration
   b) Bottom Up Integration
   c) Hybrid Integration

SOFTWARE TESTING

Integration Testing: Integration Strategies / Approaches

1. **Big-Bang Integration:**

   a) All the modules of the system are simply put together and tested. It is the simplest integration testing approach, where all the modules are combining and verifying the functionality after the completion of individual module testing.

   b) If once an error is found during the integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated. So, debugging errors reported during big bang integration testing are very expensive to fix.

Advantages:

1. It is convenient for small systems.

Disadvantages:

1. Fault Localization is difficult.
2. High risk critical modules are not isolated and tested on priority since all modules are tested at once.

Integration Testing: Integration Strategies / Approaches

2. **Incremental Approaches:**
   a) Testing is done by joining two or more ***logically related*** modules.
   b) Then the other related modules are added and tested for the proper functioning.
   c) The process continues until all of the modules are joined and tested successfully.

Different Methods of Incremental Approach are :

a) Top Down Integration
b) Bottom Up Integration
c) Hybrid Integration

Integration Testing: Integration Strategies / Approaches

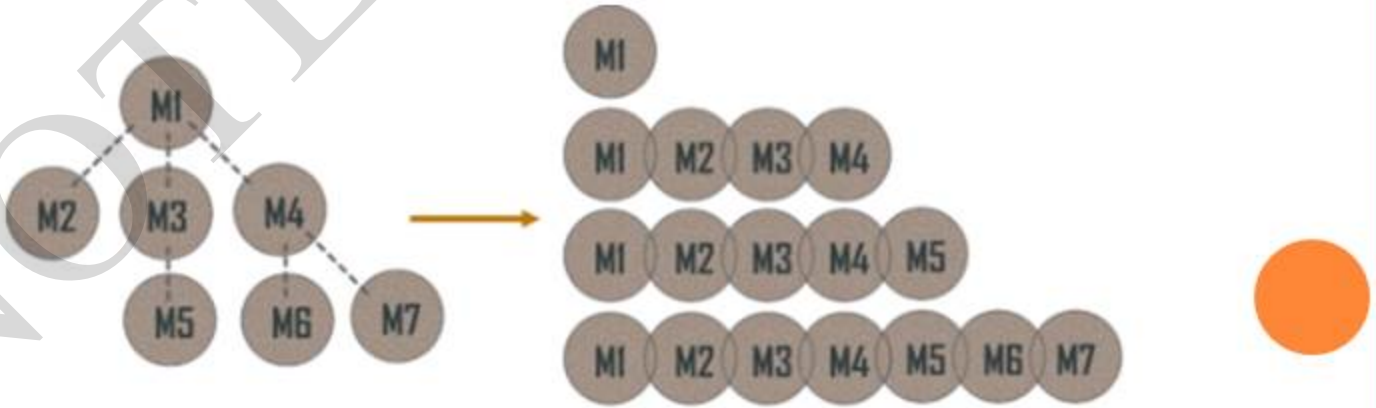
2. Incremental Approaches: Methods of Incremental Approach are :

a) Top Down Integration:

i. The top-down approach starts by testing the top-most modules and gradually moving down to the lowest set of modules one-by-one.

ii. Testing takes place from top to down following the control flow of the software system.

iii. As there is a possibility that the lower level modules might not have been developed while top modules are tested.



Integration Testing: Integration Strategies / Approaches

2. **Incremental Approaches:** Methods of Incremental Approach are :

   a) **Top Down Integration:**

   Advantages:

   i. Fault localization is easier
   ii. The test product is extremely consistent
   iii. The stubs can be written in lesser time compared to drivers
   iv. Critical modules are tested on priority
   v. Major design flaws are detected as early as possible

   Disadvantages:

   i. Requires several stubs
   ii. Poor support for early release
   iii. Basic functionality is tested at the end of the cycle

Integration Testing: Integration Strategies / Approaches

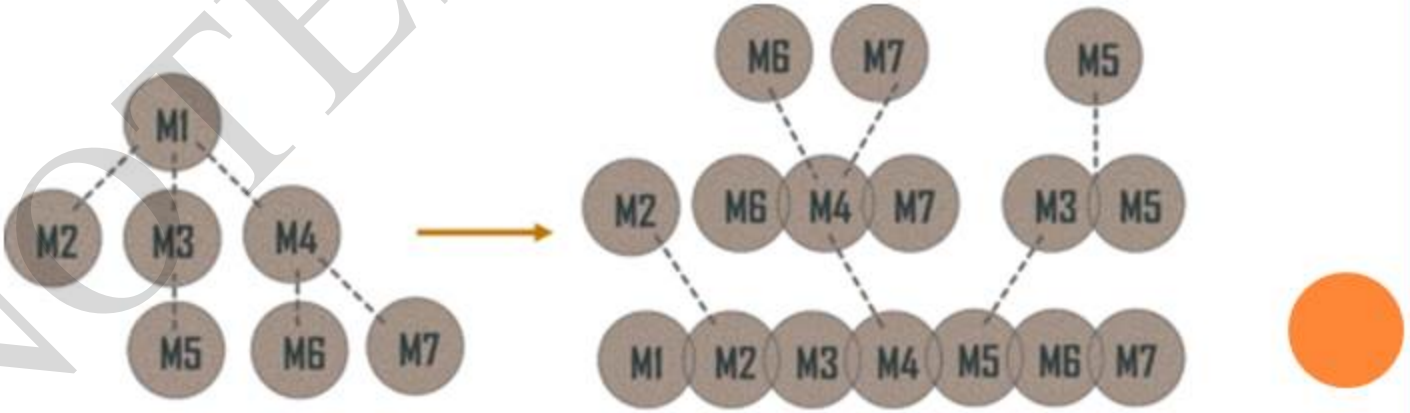
2. Incremental Approaches: Methods of Incremental Approach are :

b) Bottom Up Integration:

i. The bottom-up approach starts with testing the lowest units of the application and gradually moving up one-by-one.

ii. Here testing takes place from the bottom of the control flow to upwards.

iii. Again it's possible that the higher level modules might not have been developed by the time lower modules are tested.



Integration Testing: Integration Strategies / Approaches

2. **Incremental Approaches:** Methods of Incremental Approach are :

   b) **Bottom Up Integration:**

   Advantages:

   i. A principle advantage of this integration testing is that several disjoint subsystems can be tested simultaneously.
   ii. Test conditions are much easy to create

   Disadvantages:

   i. Requires several drivers
   ii. Data flow is tested very late
   iii. Need for drivers complicates test data management
   iv. Poor support for early release
   v. Key interfaces defects are detected late

Integration Testing: Integration Strategies / Approaches

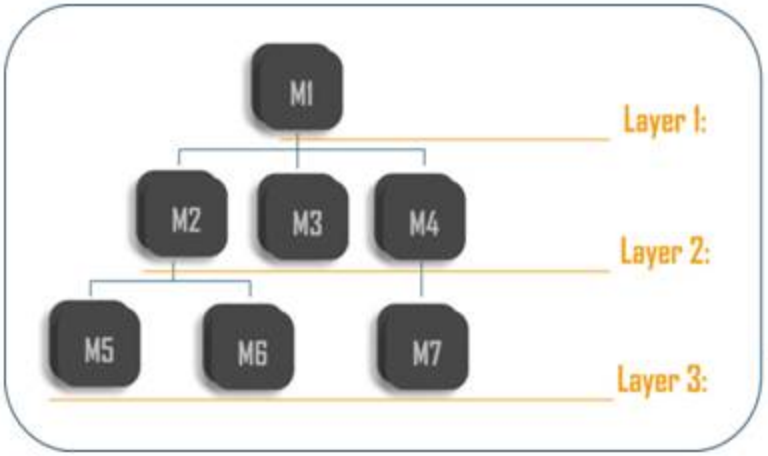
2. Incremental Approaches: Methods of Incremental Approach are :

c) Hybrid Integration:

i. To overcome the limitations and to exploit the advantages of top-down and bottom-up approaches, a hybrid approach of integration testing is used. This approach is known as sandwich integration testing or mixed integration testing.

ii. Here, the system is viewed as three layers. Main target layer in the middle, another layer above the target layer, and the last layer below the target layer.

iii. The top-down approach is used on the layer from the top to the middle layer. The bottom-up approach is used on the layer from the bottom to middle.



Integration Testing: Integration Strategies / Approaches

2. **Incremental Approaches:** Methods of Incremental Approach are :

   c) **Hybrid Integration:**

   Advantages:

   i. Top-Down and Bottom-Up testing techniques can be performed in parallel or one after the other.
   ii. Very useful for large enterprises and huge projects that further have several subprojects.

   Disadvantages:

   i. The cost requirement is very high.
   ii. Cannot be used for smaller systems with huge interdependence between the modules.
   iii. Different skill sets are required for testers at different levels.

SOFTWARE TESTING

jims Greater Noida

System Testing:

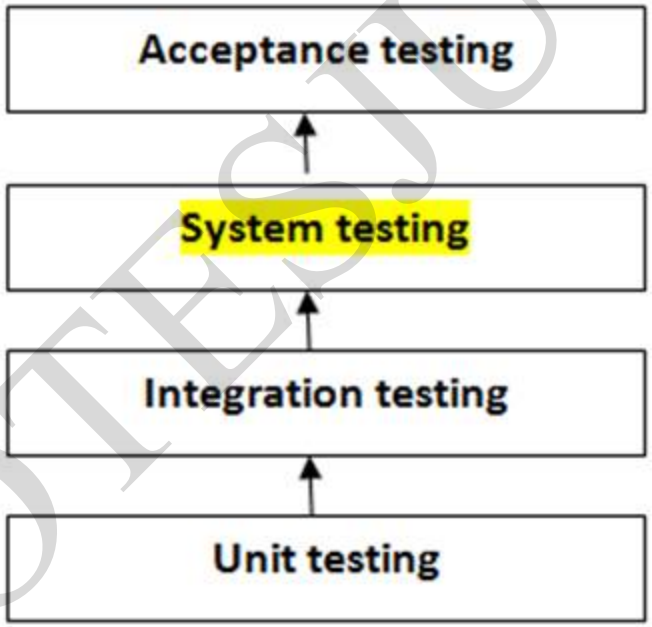
1. System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation.
2. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements.
3. This testing checks complete end-to-end scenario as per the customer's point of view.
4. Functional and Non-Functional (black box and white box testing) tests also done by System testing.
5. All things are done to maintain trust within the development that the system is defect-free and bug-free.
6. System testing is also intended to test hardware/software requirements specifications.

SOFTWARE TESTING

System Testing: Focus

System testing focuses on:

- Performance
- Security
- Recovery
- Interface
- Install-ability
- Usability
- Documentation
- Load/Stress



System Testing must be done after Unit and Integration Testing

System Testing: Performance Testing

1. Performance Testing checks the ***speed, response time, reliability, resource usage, scalability*** of a software program under their expected workload. Performance Testing is popularly called "Perf Testing"
2. Performance Testing is done to provide stakeholders with information about their application regarding speed, stability, and scalability.
3. More importantly, Performance Testing uncovers what needs to be improved before the product goes to market.

The focus of Performance Testing is checking a software program's

1. Speed : Determines whether the application responds quickly
2. Scalability: Determines maximum user load, the software application can handle.
3. Stability: Determines if the application is stable under varying loads

System Testing: Types Performance Testing

1. **Load testing** - checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
2. **Stress testing** - involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
3. **Endurance testing** - is done to make sure the software can handle the expected load over a long period of time.
4. **Spike testing** - tests the software's reaction to sudden large spikes in the load generated by users.
5. **Volume testing** - Under Volume Testing large no. of. Data is populated in a database and the overall software system's behavior is monitored. The objective is to check software application's performance under varying database volumes.
6. **Scalability testing** - The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.

jims Greater Noida

System Testing: Common Performance Problem

Long Load time - Load time is normally the initial time it takes an application to start. This should generally be kept to a minimum. While some applications are impossible to make load in under a minute, Load time should be kept under a few seconds if possible.

Poor response time - Response time is the time it takes from when a user inputs data into the application until the application outputs a response to that input. Generally, this should be very quick. Again if a user has to wait too long, they lose interest.

Poor scalability - A software product suffers from poor scalability when it cannot handle the expected number of users or when it does not accommodate a wide enough range of users. Load Testing should be done to be certain the application can handle the anticipated number of users.

Bottlenecking - Bottlenecks are obstructions in a system which degrade overall system performance. Bottlenecking is when either coding errors or hardware issues cause a decrease of throughput under certain loads. Bottlenecking is often caused by one faulty section of code. The key to fixing a bottlenecking issue is to find the section of code that is causing the slowdown and try to fix it there. Bottlenecking is generally fixed by either fixing poor running processes or adding additional Hardware.

Some **common performance bottlenecks** are

a) CPU utilization b) Memory utilization c) Network utilization

d) Operating System limitations e) Disk usage

System Testing: Error Seeding

1. Error Seeding is the process of deliberately introducing errors within a program to check whether the test cases are able to capture the seeded errors. This technique aims to detect errors in order to find out the ratio between the actual and artificial errors.
2. Artificial faults are the unknown faults and actual faults are the injected faults, hence test cases are used to check presence of such faults.
3. The reason for error seeding is that both testers and developers get a chance to challenge their respective responsibilities.
4. It is basically an estimation technique which helps to figure out the presence of real errors on the basis of the number of seeded errors found.

Debugging Activities:

1. Debugging is the process of fixing a bug in the software, it refers to identifying, analyzing and removing errors.
2. This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software.
3. It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.

Steps involved in debugging are:

1. Problem identification and report preparation.
2. Assigning the report to software engineer to the defect to verify that it is genuine.
3. Defect Analysis using modeling, documentations, finding and testing candidate flaws, etc.
4. Defect Resolution by making required changes to the system.
5. Validation of corrections.

Debugging Activities:

Debugging Strategies:

1. Study the system for the larger duration in order to understand the system. It helps debugger to construct different representations of systems to be debugging depends on the need. Study of the system is also done actively to find recent changes made to the software.
2. Backwards analysis of the problem which involves tracing the program backward from the location of failure message in order to identify the region of faulty code. A detailed study of the region is conducting to find the cause of defects.
3. Forward analysis of the program involves tracing the program forwards using breakpoints or print statements at different points in the program and studying the results. The region where the wrong outputs are obtained is the region that needs to be focused to find the defect.
4. Using the past experience of the software debug the software with similar problems in nature. The success of this approach depends on the expertise of the debugger.

Debugging Activities:

Debugging Approaches:

1. **Brute Force Method:** This method is most common and least efficient for isolating the cause of a software error. This method is applied when all else fail. In this method, a printout of all registers and relevant memory locations is obtained and studied. All dumps should be well documented and retained for possible use on subsequent problems.
2. **Back Tracking Method:** It is a quite popular approach of debugging which is used effectively in case of small applications. The process starts from the site where a particular symptom gets detected, from there on backward tracing is done across the entire source code till we are able to lay our hands on the site being the cause. Unfortunately, as the number of source lines increases, the number of potential backward paths may become unmanageably large.
3. **Cause Elimination:** The third approach to debugging, cause elimination, is manifested by induction or deduction and introduces the concept of binary partitioning. This approach is also called induction and deduction. Data related to the error occurrence are organized to isolate potential causes. A "cause hypothesis" is devised and the data are used to prove or disprove the hypothesis. Alternatively, a list of all possible causes is developed and tests are conducted to eliminated each. If initial tests indicate that a particular cause hypothesis shows promise, the data are refined in an attempt to isolate the bug.

SOFTWARE TESTING

jims Greater Noida

Difference Between Debugging and Testing

Testing	Debugging
Testing gets started with known conditions with expected results.	This is manual step by step unstructured and unreliable process to find and removes a specific bug from the system.
It performed based on the testing type which we need to perform unit testing, integration testing, system testing, user acceptance testing, stress, load, performance testing etc.	It performed based on the type of bug.
Testing is the process which can be planned, designed and executed.	Debugging is the process which cannot be so forced.
It is the process to identify the failure of implemented code.	It is the process to give the absolution to code failure.
It is a demonstration of error or apparent correctness.	It is always treated as a deductive process.
Testing is the display of errors	It is a deductive process
Design knowledge is not required for testing the system under test. Any person with or without test case can do testing.	Detailed design knowledge is definitely required to perform debugging.
Testing can be outsourced to outside team as well.	Debugging cannot be outsourced to outside team. It must be done by inside development team.
Most of the test cases in testing can be automated.	Automation in the debugging cannot possible.
Testing is the process to identify the bugs in the system under test.	Debugging is the process to identify the root cause of the bugs.

REFERENCE

Book:

1. Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007
2. An Integrated Approach to Software Engineering by Pankaj Jalote (IIT Kanpur), © 2005 by Springer Science-i-Business Media, Inc.

Website

- https://www.guru99.com
- https://www.tutorialspoint.com/software_testing_dictionary/code_review.htm
- https://www.geeksforgeeks.org/software-testing-basics/
- https://www.softwaretestingmaterial.com/software-testing/
- https://www.atlassian.com/agile/software-development/code-reviews
- https://smartbear.com/learn/code-review/what-is-code-review/
- http://softwaretestingfundamentals.com/black-box-testing/
- http://softwaretestingfundamentals.com/functional-testing/
- https://www.rainforestqa.com/blog/2016-06-27-what-is-functional-testing/
- http://softwaretestingbooks.com/functional-testing
- https://www.testbytes.net/blog/structural-testing-in-software-testing/
- https://www.edureka.co/blog/what-is-integration-testing-a-simple-guide-on-how-to-perform-integration-testing/
- https://www.softwaretestingclass.com/difference-between-testing-and-debugging/

SOFTWARE ENGINEERING

Unit 4: Software Maintenance

Shree Harsh Attri
(Assistant Professor)
Department: Bachelor of Computer Applications

JIMS Engineering Management Technical Campus
Greater Noida (U.P)

SOFTWARE MAINTENANCE

Software maintenance is a part of Software Development Life Cycle. Its main purpose is to modify and update software application after delivery to correct faults and to improve performance. Software is a model of the real world. When the real world changes, the software requires alteration wherever possible.

Need for Maintenance –
Software Maintenance must be performed in order to:

1. Correct faults
2. Improve the design
3. Implement enhancements
4. Interface with other systems
5. Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
6. Retire software.

SOFTWARE MAINTENANCE

jims Greater Noida

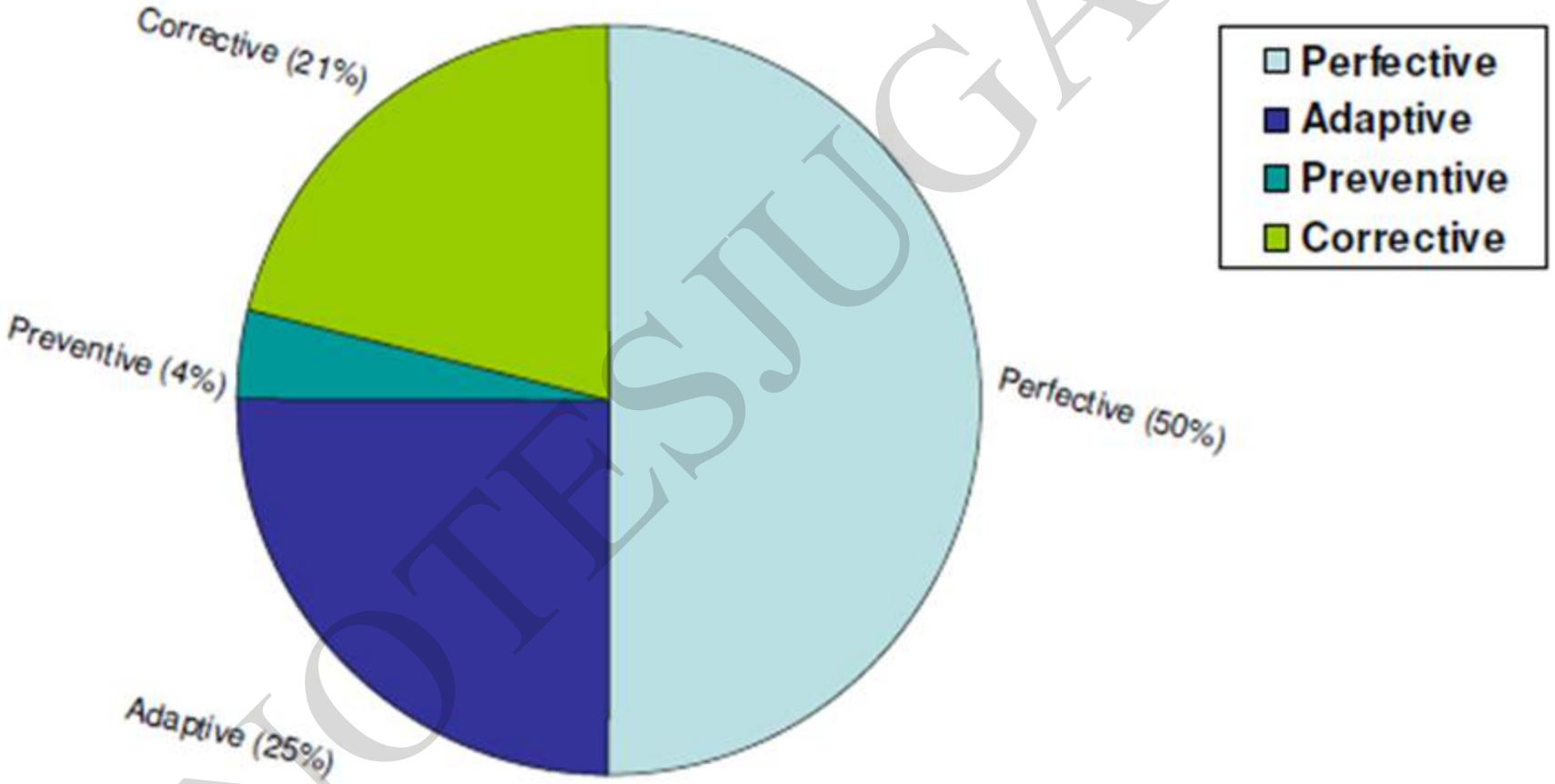
Categories of Software Maintenance:

Maintenance is divided into the following categories:

1. **Corrective Maintenance:** Corrective maintenance of a software product may be essential either to rectify some bugs observed while the system is in use, or to enhance the performance of the system.
2. **Adaptive Maintenance:** This includes modifications and updations when the customers need the product **to run on new platforms**, on new operating systems, or when they need the product to interface with new hardware and software.
3. **Perfective Maintenance:** A software product needs maintenance **to support the new features** that the users want or to change different types of functionalities of the system according to the customer demands.
4. **Preventive Maintenance:** This type of maintenance includes modifications and updations to prevent future problems of the software. It goals to attend problems, which are not significant at this moment but may cause serious issues in future.

SOFTWARE MAINTENANCE

Cost of Maintenance



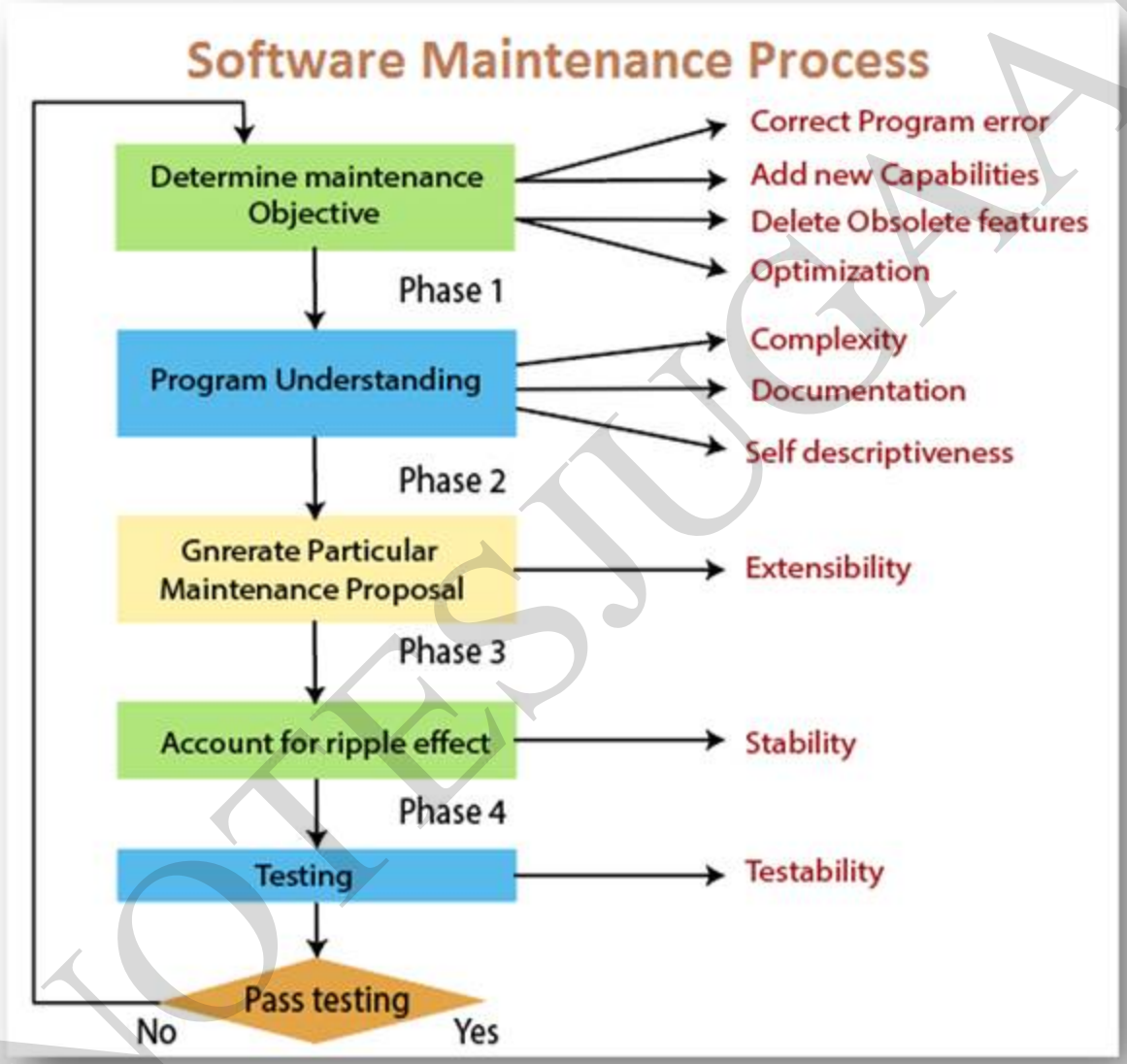
SOFTWARE MAINTENANCE

jims Greater Noida

Problems During Maintenance

- Often the program is written by another person or group of persons.
- Often the program is changed by person who did not understand it clearly.
- Program listings are not structured.
- High staff turnover.
- Information gap.
- Systems are not designed for change.

SOFTWARE MAINTENANCE PROCESS



The Software Maintenance Process

SOFTWARE MAINTENANCE PROCESS

1. **Program Understanding:** The first phase consists of analyzing the program in order to understand.

2. **Generating Particular Maintenance Proposal:** The second phase consists of generating a particular maintenance proposal to accomplish the implementation of the maintenance objective.

3. **Ripple Effect:** The third phase consists of accounting for all of the ripple effect as a consequence of program modifications.

4. **Modified Program Testing:** The fourth phase consists of testing the modified program to ensure that the modified program has at least the same reliability level as before.

5. **Maintainability:** Each of these four phases and their associated software quality attributes are critical to the maintenance process. All of these factors must be combined to form maintainability.

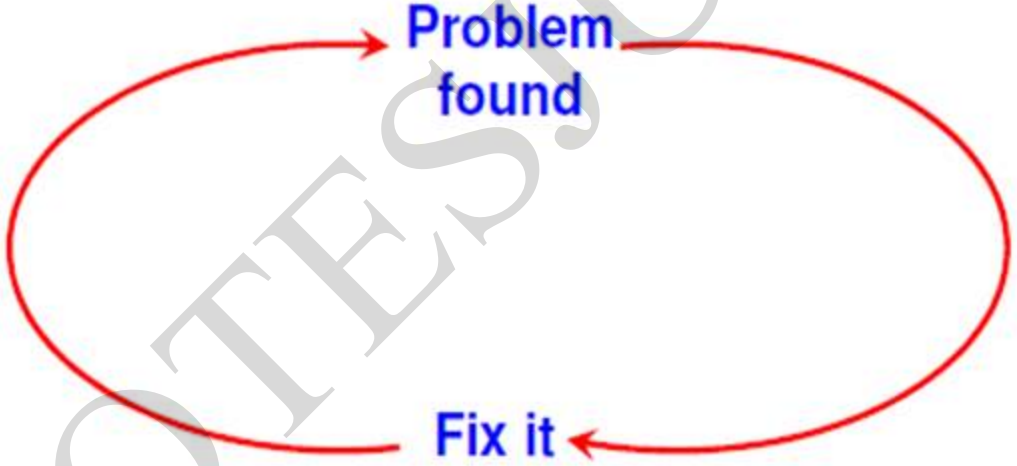
SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Maintenance Models

- Quick-fix Model

This is basically an adhoc approach to maintaining software. It is a fire fighting approach, waiting for the problem to occur and then trying to fix it as quickly as possible.



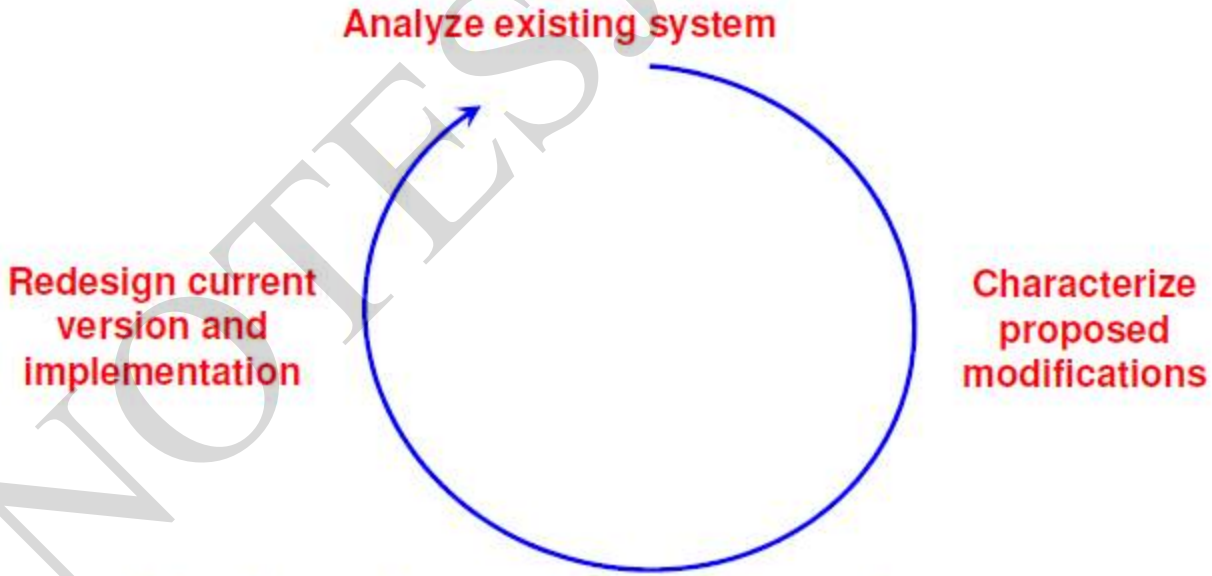
The quick-fix model

SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Maintenance Models

- Iterative Enhancement Model
  - Analysis
  - Characterization of proposed modifications
  - Redesign and implementation



The three stage cycle of iterative enhancement

SOFTWARE MAINTENANCE PROCESS

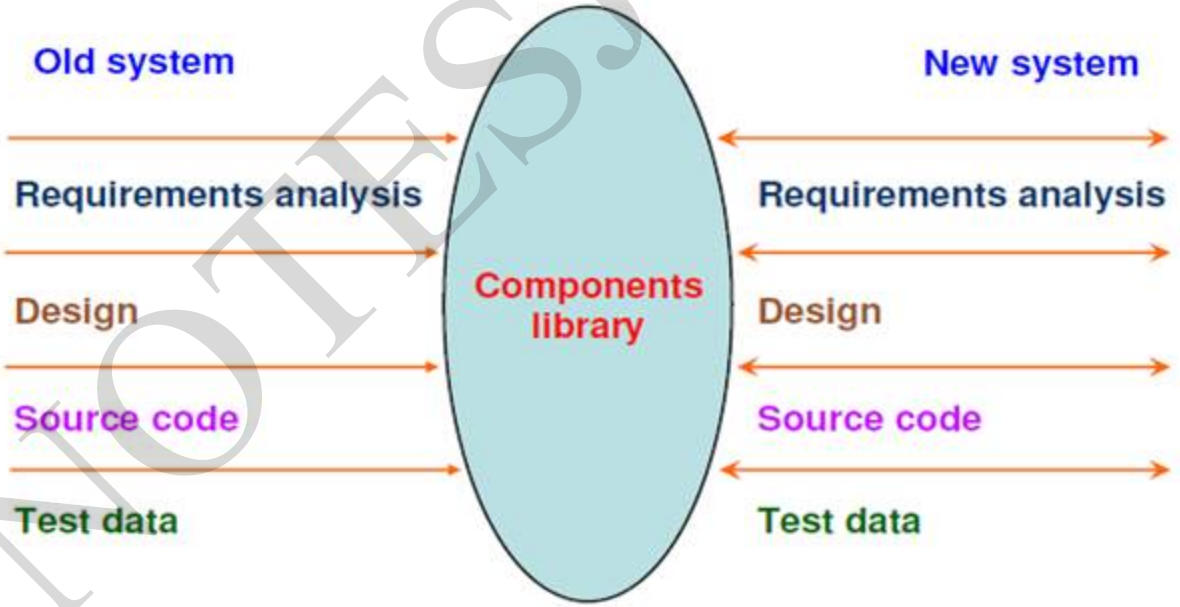
jims Greater Noida

Maintenance Models

- Reuse Oriented Model

The reuse model has four main steps:

1. Identification of the parts of the old system that are candidates for reuse.
2. Understanding these system parts.
3. Modification of the old system parts appropriate to the new requirements.
4. Integration of the modified parts into the new system.



The reuse model

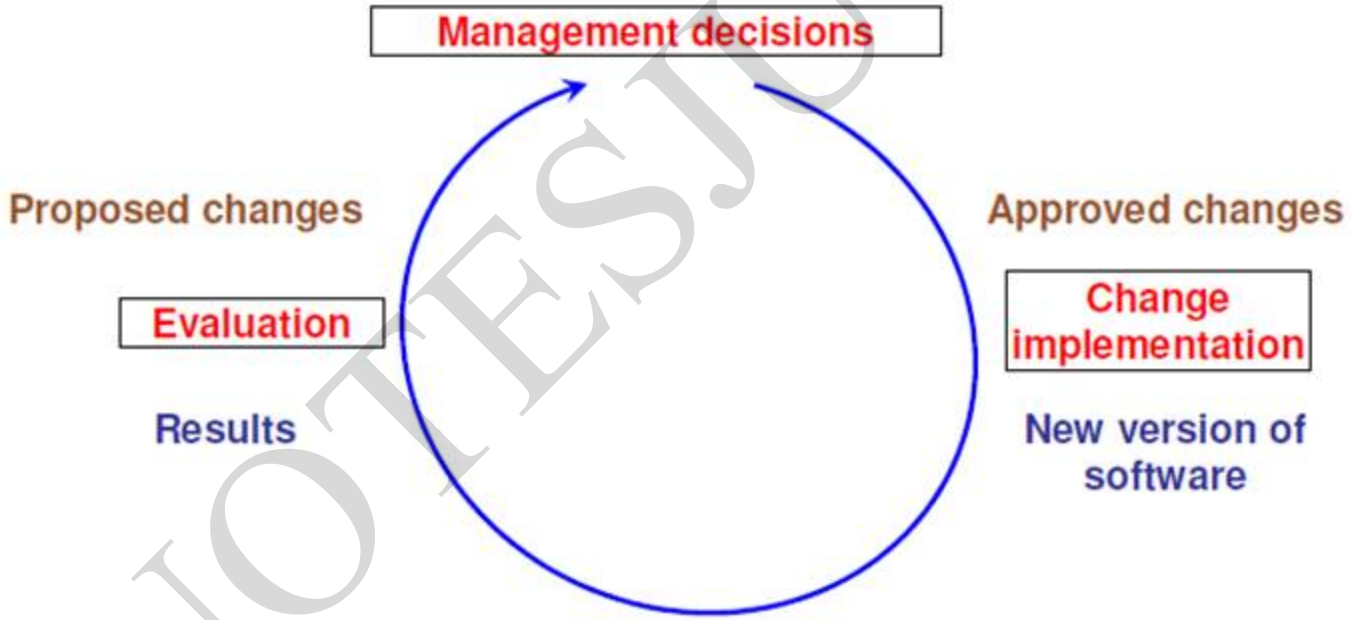
SOFTWARE MAINTENANCE PROCESS

Maintenance Models

- Boehm's Model

Boehm proposed a model for the maintenance process based upon the economic models and principles.

Boehm represent the maintenance process as a closed loop cycle.



Boehm's model

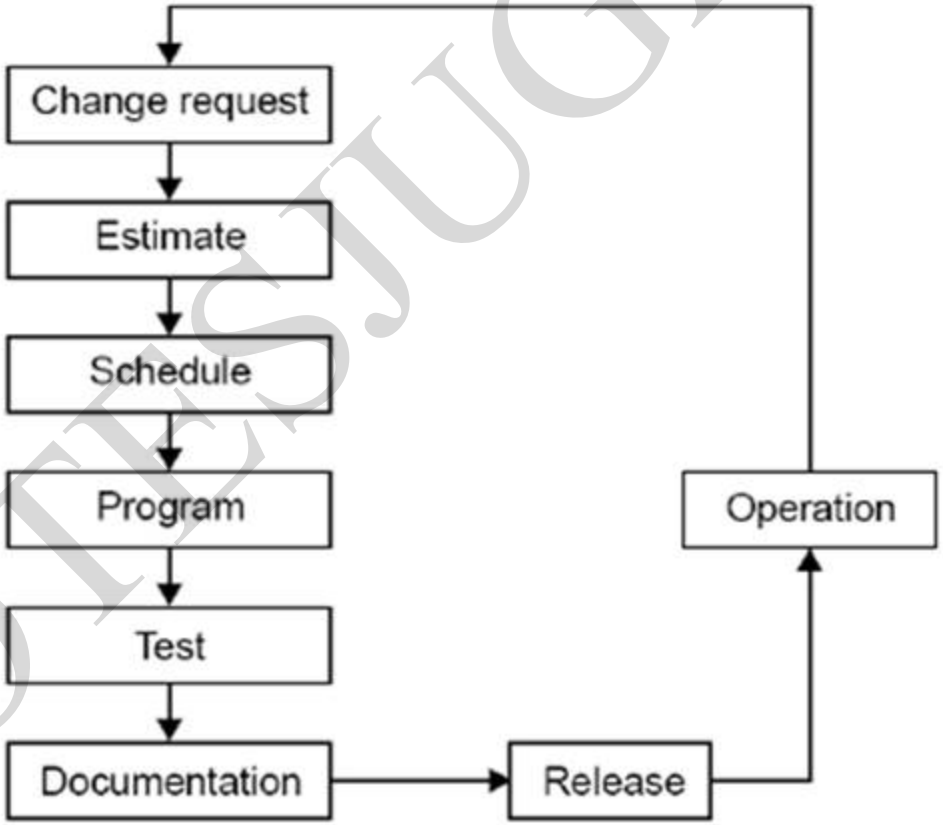
SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Maintenance Models

- Taute Maintenance Model

It is a typical maintenance model and has eight phases in cycle fashion. The phases are shown in Fig.



Taute maintenance model

Reverse Engineering

Reverse engineering is the process followed in order to find difficult, unknown and hidden information about a software system.

- Scope and Tasks

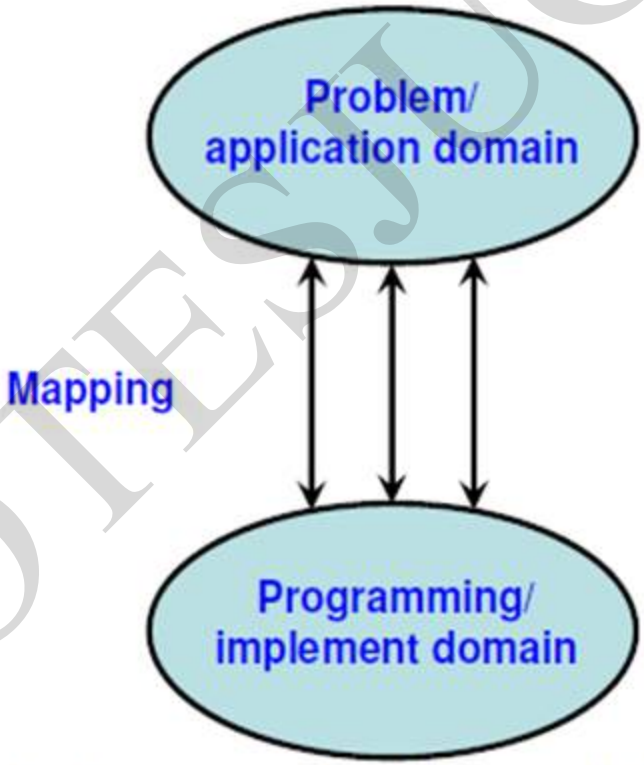
The areas there reverse engineering is applicable include (but not limited to):

1. Program comprehension
2. Redocumentation and/ or document generation
3. Recovery of design approach and design details at any level of abstraction
4. Identifying reusable components
5. Identifying components that need restructuring
6. Recovering business rules, and
7. Understanding high level system description

Reverse Engineering

Reverse Engineering encompasses a wide array of tasks related to understanding and modifying software system. This array of tasks can be broken into a number of classes.

- Mapping between application and program domains



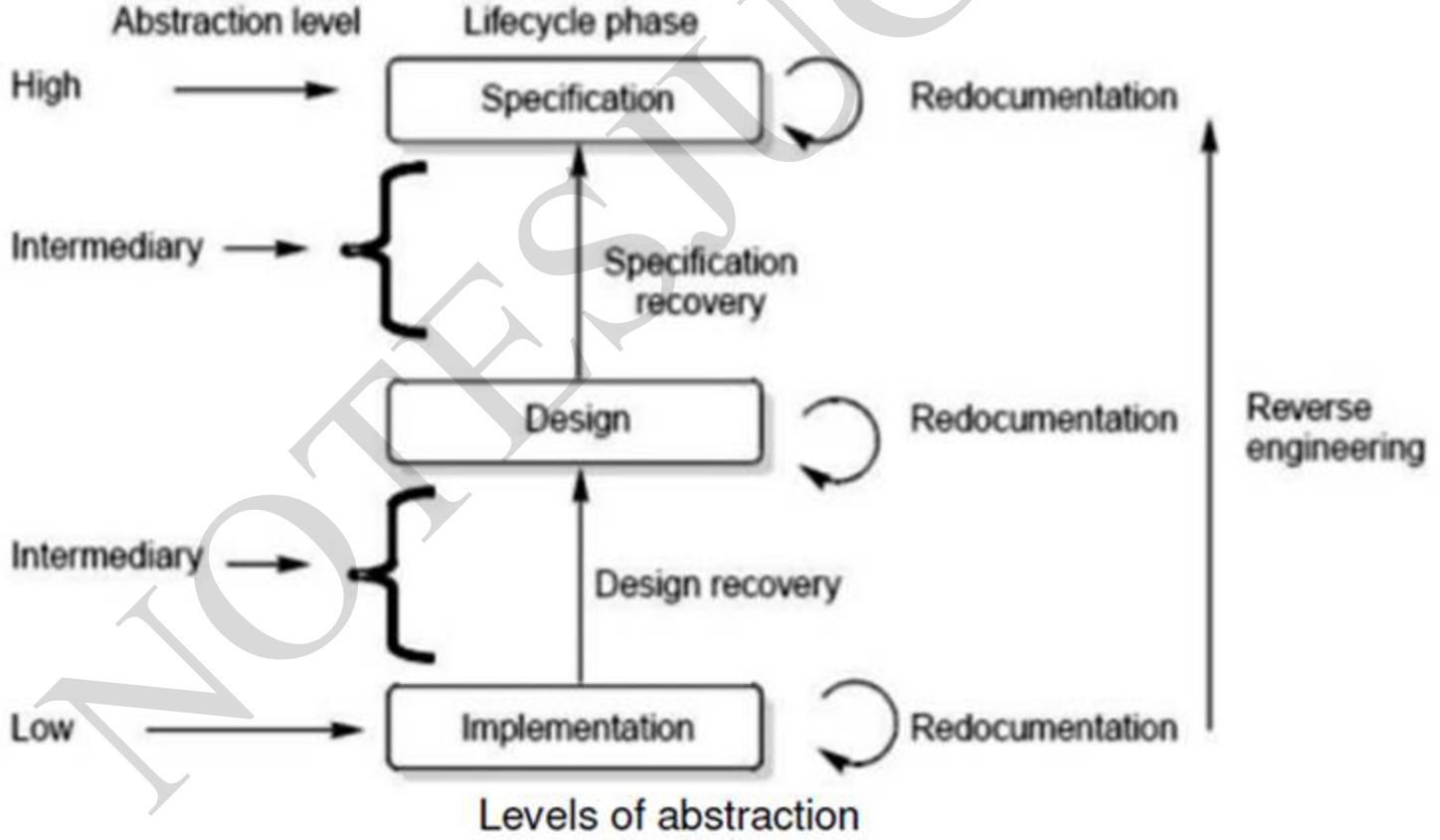
1. Mapping between concrete and abstract levels.
2. Rediscovering high level structures.
3. Finding missing links between program syntax and semantics.
4. To extract reusable component

Mapping between application and domains program

Reverse Engineering

Levels of Reverse Engineering:

1. Reverse Engineers detect low level implementation constructs and replace them with their high level counterparts.
2. The process eventually results in an incremental formation of an overall architecture of the program.



Reverse Engineering

Redocumentation

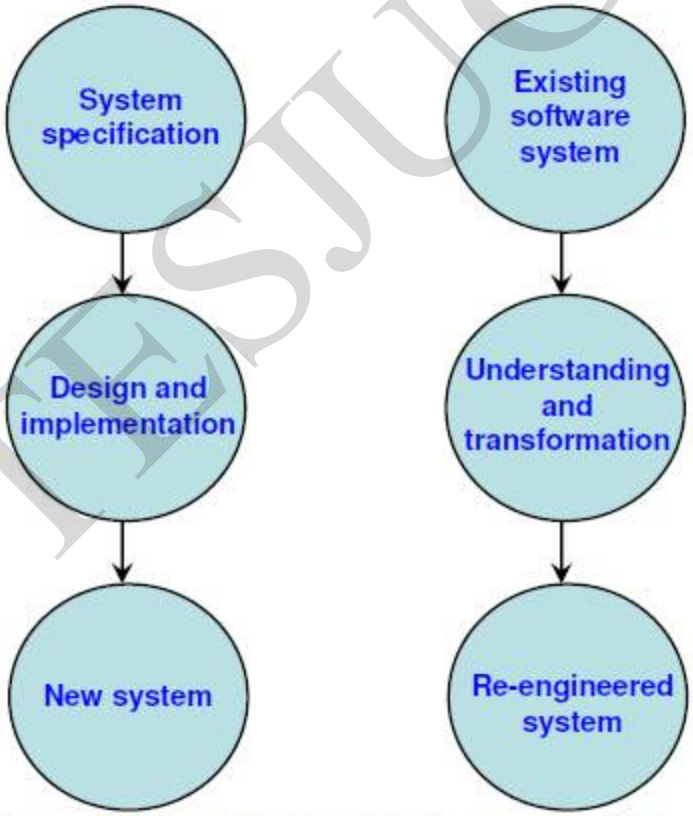
Redocumentation is the recreation of a semantically equivalent representation within the same relative abstraction level.

Design recovery

Design recovery entails identifying and extracting meaningful higher level abstractions beyond those obtained directly from examination of the source code. This may be achieved from a combination of code, existing design documentation, personal experience, and knowledge of the problem and application domains.

Software Re-Engineering

1. Software re-engineering is concerned with taking existing legacy systems and re-implementing them to make them more maintainable.
2. The critical distinction between re-engineering and new software development is the starting point for the development



Comparison of new software development with re-engineering

SOFTWARE MAINTENANCE PROCESS

Software Re-Engineering

To do modification in software under Reverse Reengineering Following suggestions

- ✓ Study code well before attempting changes
- ✓ Concentrate on overall control flow and not coding
- ✓ Heavily comment internal code
- ✓ Create Cross References
- ✓ Build Symbol tables
- ✓ Use own variables, constants and declarations to localize the effect
- ✓ Keep detailed maintenance document
- ✓ Use modern design techniques

SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Software Re-Engineering

To do modification in software under Reverse Reengineering Following suggestions

- Source Code Translation

1. **Hardware platform update:** The organization may wish to change its standard hardware platform. Compilers for the original language may not be available on the new platform.

2. **Staff Skill Shortages:** There may be lack of trained maintenance staff for the original language. This is a particular problem where programs were written in some non standard language that has now gone out of general use.

3. **Organizational policy changes:** An organization may decide to standardize on a particular language to minimize its support software costs. Maintaining many versions of old compilers can be very expensive.

Software Re-Engineering

Program Restructuring

1. **Control flow driven restructuring:** This involves the imposition of a clear control structure within the source code and can be either inter modular or intra modular in nature.
2. **Efficiency driven restructuring:** This involves restructuring a function or algorithm to make it more efficient. A simple example is the replacement of an IF-THEN-ELSE-IF-ELSE construct with a CASE construct.
3. **Adaption driven restructuring:** This involves changing the coding style in order to adapt the program to a new programming language or new operating environment, for instance changing an imperative program in PASCAL into a functional program in LISP.

```
IF Score > = 75 THEN Grade: = 'A'
ELSE IF Score > = 60 THEN Grade: = 'B'
ELSE IF Score > = 50 THEN Grade: = 'C'
ELSE IF Score > = 40 THEN Grade: = 'D'
ELSE IF Grade = 'F'
END
```

(a)

```
CASE Score of
75, 100: Grade: = 'A'
60, 74: Grade: = 'B';
50, 59: Grade: = 'C';
40, 49: Grade: = 'D';
ELSE Grade: = 'F'
END
```

(b)

Restructuring a program

SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Configuration Management

The process of software development and maintenance is controlled is called configuration management. The configuration management is different in development and maintenance phases of life cycle due to different environments.

- Configuration Management Activities

The activities are divided into four broad categories.

1. The identification of the components and changes
2. The control of the way by which the changes are made
3. Auditing the changes
4. Status accounting recording and documenting all the activities that have take place

SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Configuration Management

The following documents are required for these activities

- ✓ Project plan
- ✓ Software requirements specification document
- ✓ Software design description document
- ✓ Source code listing
- ✓ Test plans / procedures / test cases
- ✓ User manuals

SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Configuration Management Activities

1. **Configuration Identification:** Configuration identification is a method of determining the scope of the software system. It is a description that contains the CSCI type (Computer Software Configuration Item), a project identifier and version information.

Activities during this process:

a) Identification of configuration items like source code modules, test case, and requirements specification.
b) Identification of each CSCI in the **S**ource **C**onfiguration **M**anagement (SCM) repository, by using an object-oriented approach
c) The process starts with basic objects which are grouped into aggregate objects. Details of what, why, when and by whom changes in the test are made
d) Every object has its own features that identify its name that is explicit to all other objects
e) List of resources required such as the document, the file, tools, etc.

Example:
Instead of naming a File login.php its should be named login_v1.2.php where v1.2 stands for the version number of the file

SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Configuration Management Activities

2. **Change Control:** Change control is a procedural method which ensures quality and consistency when changes are made in the configuration object. In this step, the change request is submitted to software configuration manager.

Activities during this process:

a) Control ad-hoc change to build stable software development environment. Changes are committed to the repository
b) The request will be checked based on the technical merit, possible side effects and overall impact on other configuration objects.
c) It manages changes and making configuration items available during the software lifecycle

SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Configuration Management Activities

3. **Configuration Audits and Reviews:** Software Configuration audits verify that all the software product satisfies the baseline needs. It **ensures that what is built is what is delivered**.

Activities during this process:

a) Configuration auditing is conducted by auditors by checking that defined processes are being followed and ensuring that the SCM goals are satisfied.
b) To verify compliance with configuration control standards. auditing and reporting the changes made.
c) SCM audits also ensure that traceability is maintained during the process.
d) Ensures that changes made to a baseline comply with the configuration status reports.
e) Validation of completeness and consistency.

SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Configuration Management Activities

4. **Configuration Status Accounting:** Configuration status accounting tracks each release during the SCM process. This stage involves tracking what each version has and the changes that lead to this version.

Activities during this process:

a) Keeps a record of all the changes made to the previous baseline to reach a new baseline.
b) Identify all items to define the software configuration.
c) Monitor status of change requests.
d) Complete listing of all changes since the last baseline .
e) Allows tracking of progress to next baseline.
f) Allows to check previous releases/versions to be extracted for testing.

SOFTWARE MAINTENANCE PROCESS

jims Greater Noida

Documentation

Software documentation is the written record of the facts about a software system recorded with the intent to convey purpose, content and clarity.

- User Documentation

S.No.	Document	Function
1.	System Overview	Provides general description of system's functions.
2.	Installation Guide	Describes how to set up the system, customize it to local hardware needs and configure it to particular hardware and other software systems.
3.	Beginner's Guide	Provides simple explanations of how to start using the system.
4.	Reference Guide	Provides in depth description of each system facility and how it can be used.
5.	Enhancement	Booklet Contains a summary of new features.
6.	Quick reference card	Serves as a factual lookup.
7.	System administration	Provides information on services such as net-working, security and upgrading.

User Documentation

SOFTWARE MAINTENANCE PROCESS

- System Documentation

It refers to those documentation containing all facets of system, including analysis, specification, design, implementation, testing, security, error diagnosis and recovery.

- System Documentation

S.No.	Document	Function
1.	System Rationale	Describes the objectives of the entire system.
2.	SRS	Provides information on exact requirements of system as agreed between user and developers.
3.	Specification/ Design	Provides description of: (i) How system requirements are implemented. (ii) How the system is decomposed into a set of interacting program units. (iii) The function of each program unit.
4.	Implementation	Provides description of: (i) How the detailed system design is expressed in some formal programming language. (ii) Program actions in the form of intra program comments.

SOFTWARE MAINTENANCE PROCESS

- System Documentation

S.No.	Document	Function
5.	System Test Plan	Provides description of how program units are tested individually and how the whole system is tested after integration.
6.	Acceptance Test Plan	Describes the tests that the system must pass before users accept it.
7.	Data Dictionaries	Contains description of all terms that relate to the software system in question.

System Documentation

REFERENCE

Book:

1. Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007
2. An Integrated Approach to Software Engineering by Pankaj Jalote (IIT Kanpur), © 2005 by Springer Science-i-Business Media, Inc.

Website

https://www.geeksforgeeks.org/software-engineering-software-maintenance/

https://economictimes.indiatimes.com/definition/software-maintenance