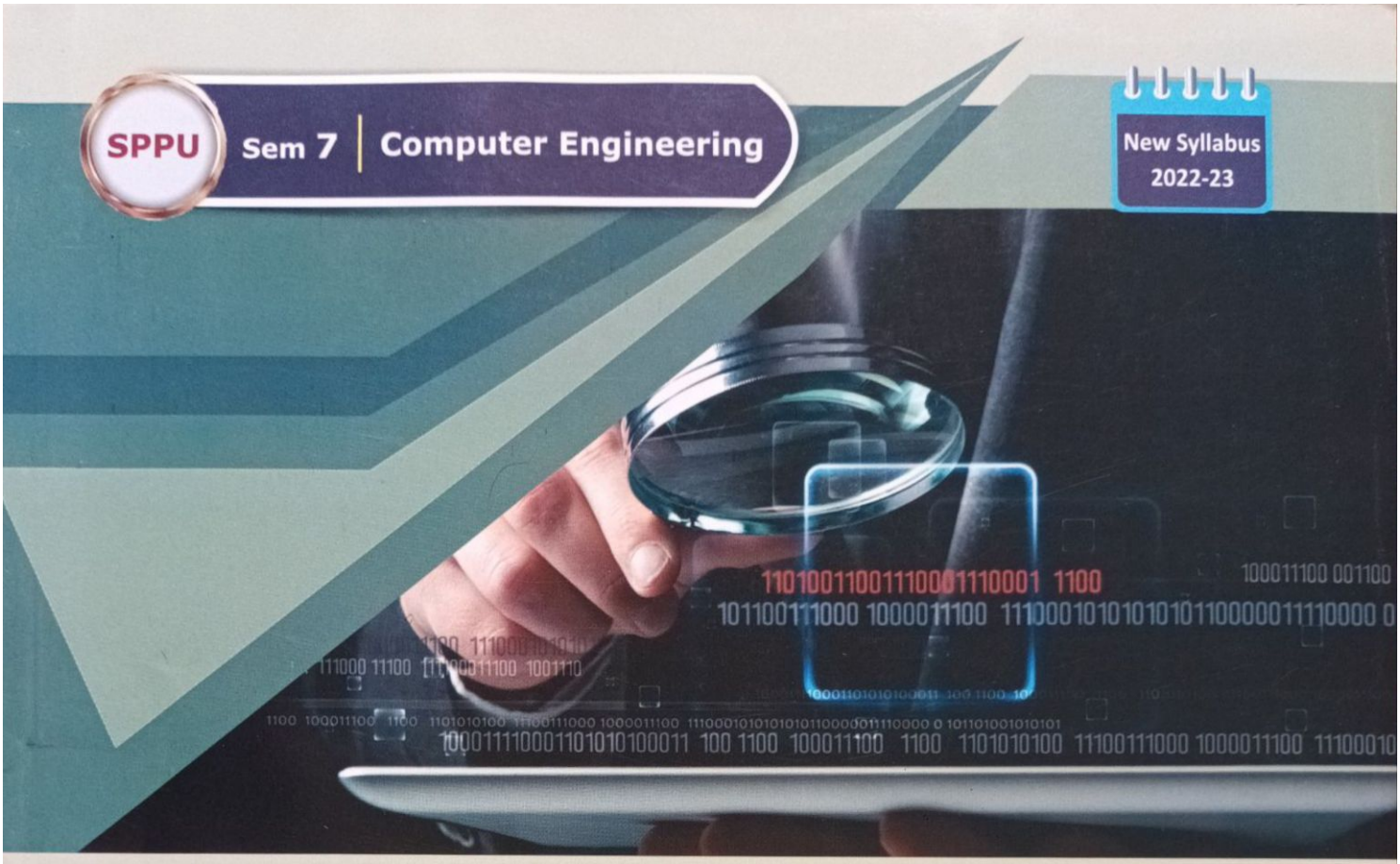
SPPU | Sem 7 | Computer Engineering

New Syllabus 2022-23



Strictly as per the New Syllabus (REV-2019 'C' Scheme) of Savitribai Phule Pune University w.e.f. academic year 2022-2023

(Elective IV)(Course Code 410245 (D))

Software Testing and Quality Assurance

Dr. Pankaj Agarkar
Dr. D. Y. Patil Knowledge City, Charholi (Bk), Lohgaon, Pune

Yogesh Mali
Marathwada Mitramandal's Institute of Technology (MMIT) Lohgaon, Pune

ISBN : 978-93-5583-214-6

TECH-NEO PUBLICATIONS
Where Authors Inspire Innovation
A Sachin Shah Venture

www.techneobooks.in
info@techneobooks.in

P7-80A

Price ₹ 395/-

Savitribai Phule Pune University (SPPU)

Software Testing and Quality Assurance

[Course Code : 410245 (D)] (Elective IV)

Semester 7 - Computer Engineering

Strictly as per the New Syllabus (2019 Course) of
Savitribai Phule Pune University w.e.f. Academic Year 2022-2023

Dr. Pankaj Agarkar

BE (CE), ME(CE),
PhD (Computer Engineering)
LMISTE, PGACDE, MBA (AIEM)
HOD Computer Department, Affiliated to Pune University
Dr. D. Y. Patil School of Engineering
Dr. D. Y. Patil Technical Campus
Dr. D. Y. Patil Knowledge City, Charholi (Bk), Lohgaon, Pune, 412 105

Prof. Yogesh Mali

M.E. Computer Engineering
Assistant Professor,
Computer Engineering Department,
Marathwada Mitramandal's Institute of Technology (MMIT)
Lohgaon, Pune - 411 047

TECH-NEO PUBLICATIONS
Where Authors Inspire Innovation
A Sachin Shah Venture

P7-80

Rohit Patil

Unit II : Test Planning and Quality Management (07 Hours)

Test Planning - Artifacts, Strategy, Test Organization –Test Manager & Tester Role, Test plan purpose & contents, Test Strategy and Approach, Test cases & Test Data, Test Entry-Exit criteria, Test Execution Schedule, Use case Testing, Scenario Testing, Test Monitoring & Control- Test Metrics –Test Case Productivity, Test case Coverage, Defect Acceptance & Rejection, Test Efficiency, Efforts and Schedule Variance, Test Efforts biasing Factors, Test Report & configuration Management, Quality Assurance Process, Documentation Risk & Issues. Software Quality, Quality Management Importance, Quality Best practices. **(Refer chapter 2)**

#Exemplar/Case Studies	1. Online Recommendation System 2. Quality Engineering services for Medical Devices company Case Study (cigniti.com)

End Sem Exam 70 Marks (Unit III, IV, V, VI)

Unit III : Test Case Design Techniques (07 Hours)

Software Testing Methodologies : White Box Testing, Black Box Testing, Grey Box Testing. Test Case Design Techniques: Static Techniques: Informal Reviews, Walkthroughs, Technical Reviews, Inspection. Dynamic Techniques: Structural Techniques: Statement Coverage Testing, Branch Coverage Testing, Path Coverage Testing, Conditional Coverage Testing, Loop Coverage Testing Black Box Techniques: Boundary Value Analysis, Equivalence Class Partition, State Transition Technique, Cause Effective Graph, Decision Table, Use Case Testing, Experienced Based Techniques: Error guessing, Exploratory testing

Levels of Testing : Functional Testing: Unit Testing, Integration Testing, System Testing, User Acceptance Testing, Sanity/Smoke Testing, Regression Test, Retest. Non-Functional Testing: Performance Testing, Memory Test, Scalability Testing, Compatibility Testing, Security Testing, Cookies Testing, Session Testing, Recovery Testing, Installation Testing, Adhoc Testing, Risk Based Testing, I18N Testing, L1ON Testing, Compliance Testing.

(Refer chapter 3)

#Exemplar/Case Studies	1. Case Study: Manual Testing (Online Marketing SoftwarePlatform) Link: https://www.360logica.com/blog/case-study- manual-testing-online-marketing-software- platform/ 2. Case Study: Decision Table Testing (transferring money online to an account which is already added and approved.)

Unit IV : Software Quality Assurance and Quality Control (07 Hours)

Software Quality Assurance : Introduction, Constraints of Software Product Quality Assessment, Quality and Productivity Relationship, Requirements of a Product, Characteristics of Software, Software Development Process, Types of Products, Schemes of Criticality Definitions, Software Quality Management, Why Software Has Management System, Important Aspects of Quality Management.

Software Quality Control : Software quality models, Quality measurement and metrics, Quality plan, implementation and documentation, Quality tools including CASE tools, Quality control and reliability of quality process, Quality management system models, Complexity metrics and Customer Satisfaction, International quality standards – ISO, CMM.

(Refer chapter 4)

#Exemplar/Case Studies	1. Case Study #1 - Android Application Acceptance Test Suite 2. Case Study #2 - API Acceptance Test Suite Link for above case studies - Software Quality Assurance Case Studies - Beta Breakers

Unit V : Automation Testing Tools / Performance Testing Tools (07 Hours)

Automation Testing : What is automation testing, Automated Testing Process, Automation Frameworks, Benefits of automation testing, how to choose automation testing tools. Selenium Automation Tools: Selenium's Tool Suite- Selenium IDE, Selenium RC, Selenium Web driver, Selenium Grid. Automation Tools: SoapUI, Robotic Process Automation (RPA), Tosca, Appium. **(Refer chapter 5)**

#Exemplar/CaseStudies	1. Case Study : Cucumber open-source automation testing framework. 2. Case Study : (PDF) Automated Software Testing – A Case Study (researchgate.net)

Unit VI : Testing Framework (07 Hours)

Testing Framework : Software Quality, Software Quality Dilemma, Achieving Software Quality, Software Quality Assurance Elements of SQA, SQA Tasks, Goals and Metrics, Formal Approaches to SQA, Statistical Software Quality Assurance, Six Sigma for Software Engineering, ISO 9000 Quality Standards, SQA Plan, Total Quality Management, Product Quality Metrics, In process Quality Metrics, Software maintenance, Ishikawa's 7 basic tools, Flow Chart, Checklists, Pareto diagrams, Histogram, Run Charts, Scatter diagrams, Control chart, Cause Effect diagram. Defect Removal Effectiveness and Process. **(Refer chapter 6)**

#Exemplar/CaseStudies	1. Case study : Software Quality In Academic Curriculum. 2. Case study : Evaluation of an Automated Testing Framework : A Case Study (scielo.sa.cr)

❑❑❑

Index

In Sem

End Sem

UNIT III

CHAPTER 3

Test Case Design Techniques

University Prescribed Syllabus for the Academic Year 2022-2023

Software Testing Methodologies : White Box Testing, Black Box Testing, Grey Box Testing. Test Case Design Techniques : Static Techniques : Informal Reviews, Walkthroughs, Technical Reviews, Inspection. Dynamic Techniques: Structural Techniques: Statement Coverage Testing, Branch Coverage Testing, Path Coverage Testing, Conditional Coverage Testing, Loop Coverage Testing Black Box Techniques: Boundary Value Analysis, Equivalence Class Partition, State Transition Technique, Cause Effective Graph, Decision Table, Use Case Testing, Experienced Based Techniques: Error guessing, Exploratory testing

Levels of Testing : Functional Testing: Unit Testing, Integration Testing, System Testing, User Acceptance Testing, Sanity/Smoke Testing, Regression Test, Retest. Non-Functional Testing: Performance Testing, Memory Test, Scalability Testing, Compatibility Testing, Security Testing, Cookies Testing, Session Testing, Recovery Testing, Installation Testing, Adhoc Testing, Risk Based Testing, I18N Testing, L1ON Testing, Compliance Testing.

3.1 SOFTWARE TESTING METHODOLOGIES

GQ. Explain Software Testing Methodologies.

- Methodologies can be considered as the set of testing mechanisms used in the software development lifecycle from Unit Testing to System Testing. Selecting an appropriate testing methodology is considered to be the core of the testing process.
- With the means of security, compatibility, and usability, a software product should be tested by using the proper testing methodology.
- Basically, there are 3 testing techniques that are used for testing. They are White Box Testing, Black Box Testing, and Grey Box Testing. Each of the testing techniques is briefed below for your better understanding.

3.1.1 White Box Testing?

The white box testing technique is used to examine the program structure and business logic, it validates the code or program of an application. It is also called *Clear Box Testing, Glass Box Testing, or Open Box Testing*.

White Box Testing Techniques include :

- **Statement Coverage :** Examines all the programming statements.
- **Branch Coverage :** Series of running tests to ensure if all the branches are tested.
- **Path Coverage :** Tests all the possible paths to cover each statement and branch.

3.1.2 Black Box Testing

- The Black Box testing method is used to test the functionality of an application based on the requirement specification.
- Unlike White Box Testing it does not focus on the internal structure/code of the application.

Black Box Techniques include

(1) Boundary Value analysis
(2) Equivalence Partitioning (Equivalence Class Partitioning)
(3) Decision Tables
(4) Domain Tests
(5) State Models
(6) Exploratory Testing (Requires less preparation and also helps to find the defects quickly).

3.1.3 Grey Box Testing

- This method of testing is performed with less information about the internal structure of an application.
- Generally, this is performed like Black Box Testing only but for some critical areas of application, White Box Testing is used.

3.2 TEST CASE DESIGN TECHNIQUES

GQ. Explain Test Case Design Techniques with example.

- The most significant and crucial phase in the development of software is its testing phase.
- Not only does testing helps to determine the quality of a product but it also allows one to modify and upgrade the product in terms of end-user friendliness and usability.
- In this article, we will address the fundamental notion of test case design techniques of various kinds.
- Test cases are the fundamental building blocks which when put together form the testing phase.

GQ. What is a Test Case ?

- They are often a pre-defined set of instructions addressing the steps to be taken in order to determine whether or not the end product exhibits the desired outcome. These instructions may include predefined sets of inputs, conditions along with their respective end results.
- However, in order to be through with one's testing one could often end up with too many test cases. To avoid such scenarios, one should find the best test case design technique as per ones requirement so as to reduce a significant number of test cases.
- These test case design techniques help create effective test cases covering the various features that determine the quality and value of a product.

3.2.1 Static Techniques

- Static testing is a software testing method that involves the examination of a program, along with any associated documents, but does not require the program to be executed.
- Dynamic testing, the other main category of software testing, involves interaction with the program while it runs. The two methods are frequently used together to try to ensure the basic functionalities of a program.
- Instead of executing the code, static testing is a process of checking the code and designing documents and requirements before it's run in order to find errors.
- The main goal is to find flaws in the early stages of development. It is normally easier to find the sources of possible failures this way.

GQ. What is subject to static testing ?

- It's common for code, design documents and requirements to be static tested before the software is run in order to find errors.
- Anything that relates to functional requirements can also be checked.
- More specifically, the process will involve reviewing written materials that, altogether, give a wider view of the tested software application as a whole.
- Some examples of what's tested include :
  - Requirement specifications
  - User documents
  - Source code
  - User documents
  - Design documents
  - Web page content
  - Test cases, test data and test scripts
  - Specification and matrix documents

Benefits of static testing

Some benefits of static testing include :

- Early detection and correction of any coding errors.
- Reduces cost in early stages of development -- in terms of the amount of rework needed to fix any errors.
- Reduced timescales for development.
- Feedback received in this stage will help improve the overall functioning of the software. Once other testing types like dynamic testing start, there won't be as many errors found. This means code is overall more maintainable.
- This process will also help give developers a better idea of the quality issues found in the software.
- With automated tools, this process can be quite fast to review code and other documents.
- Static testing can also boost the amount of communication between teams.

Static testing techniques

- Static testing is carried out with two different steps or techniques -- review and static analysis.
- Static review is typically carried out to find and remove errors and ambiguities found in supporting documents. Documents reviewed include software requirements specifications, design and test cases.
- The documents can be reviewed in multiple ways, such as in a walkthrough, peer review or as an inspection.
- The next step, static analysis, is where the code written by developers is analysed. The evaluation is done in order to find any structural defects that may lead to errors when run.
- Some other techniques used while performing static testing include use case requirements validation, functional requirement validation, architecture review and field dictionary validation.
- Use case requirements ensure possible end-user actions are properly defined.
- Functional requirements will identify any necessary requirements for the software. Reviews of architecture means business-level processes are analysed. Field dictionary validation will analyse UI fields.
- Static testing may also be conducted either manually or through automation with the use of various software testing tools.

Types of static testing reviews

Reviews, the first step in static testing, can be conducted in numerous ways. Reviews are performed to find and remove errors found in supporting documents. This process can be carried out in four different ways :

1. **Informal :** Informal reviews will not follow any specific process to find errors. Co-workers can review documents and provide informal comments.
2. **Walkthrough :** The author of whichever document is being reviewed will explain the document to their team. Participants will ask questions, and any notes are written down.
3. **Inspection :** A designated moderator will conduct a strict review as a process to find defects.
4. **Technical/peer reviews :** Technical specifications are reviewed by peers in order to detect any errors.

Differences between static testing vs. dynamic testing

- Dynamic testing is a method of assessing the feasibility of a software program by giving input and examining output. The dynamic method requires that the code be compiled and run.
- The biggest and most notable difference between static and dynamic testing is that dynamic testing requires code to be executed. Functional behaviour and performance are checked to confirm if the code works properly.
- Static testing will analyse the code, requirement documents and design documents, while dynamic testing will look at the functional behaviour of software systems like memory usage and performance.
- Static testing essentially gives an assessment of code, while dynamic testing will try and find active bugs. The cost of finding code defects in dynamic testing will normally cost more in terms of time and money than in static testing.
- The two types of testing are meant not meant to be mutually exclusive, however. Ideally, they should be used together. Static testing is about the prevention of defects, whereas dynamic testing is about finding active defects.
- Types of dynamic testing include unit testing, integration testing, system testing and performance testing.

3.2.2 Informal Reviews

GQ. What is informal review in software testing ?

- Informal review is **an informal static test technique performed on any kind of requirements, design, code, or project plan**.
- During informal review, the work product is given to a domain expert and the feedback/comments are reviewed by the owner/author.
- Informal reviews are applied many times during the early stages of the life cycle of the document.
- A two person team can conduct an informal review. In later stages these reviews often involve more people and a meeting.
- The goal is to keep the author and to improve the quality of the document. The most important thing to keep in mind about the informal reviews is that they are **not documented.**

Informal reviews

- **Informal reviews** take place between two or three people.
- The review conference is scheduled at their convenience.
  - This meeting is generally scheduled during the free time of the team members.
  - There is no planning for the meeting.
  - If any errors occur, they are not corrected in the informal reviews.
  - There is no guidance from the team.
  - This review is less effective compared to the formal review.

3.2.3 Walkthrough

- Walkthrough is a method of conducting informal group/individual review. In a walkthrough, author describes and explain work product in a informal meeting to his peers or supervisor to get feedback. Here, validity of the proposed solution for work product is checked.
- It is cheaper to make changes when design is on the paper rather than at time of conversion. Walkthrough is a static method of quality assurance. Walkthrough are informal meetings but with purpose.

GQ. What are Differences between Inspection and Walkthrough ?

Sr. No.	Inspection	Walkthrough
1.	It is formal.	It is informal.
2.	Initiated by project team.	Initiated by author.
3.	A group of relevant persons from different departments participate in the inspection.	Usually team members of the same project take participation in the walkthrough. Author himself acts walkthrough leader.
4.	Checklist is used to find faults.	No checklist is used in the walkthrough.
5.	Inspection processes includes overview, preparation, inspection, and rework and follow up.	Walkthrough process includes overview, little or no preparation, little or no preparation examination (actual walkthrough meeting), and rework and follow up.
6.	Formalized procedure in each step.	No formalized procedure in the steps.

Sr. No.	Inspection	Walkthrough
7.	Inspection takes longer time as list of items in checklist is tracked to completion.	Shorter time is spent on walkthrough as there is no formal checklist used to evaluate program.
8.	Planned meeting with the fixed roles assigned to all the members involved.	Unplanned
9.	Reader reads product code. Everyone inspects it and comes up with detects.	Author reads product code and his teammate comes up with the defects or suggestions.
10.	Recorder records the defects.	Author make a note of defects and suggestions offered by teammate.
11.	Moderator has a role that moderator making sure that the discussions proceed on the productive lines.	Informal, so there is no moderator.

3.2.4 Technical Reviews

GQ. What is Technical review in software testing ?

- It is less formal review
- It is led by the trained moderator but can also be led by a technical expert
- It is often performed as a peer review without management participation
- Defects are found by the experts (such as architects, designers, key users) who focus on the content of the document.
- In practice, technical reviews vary from quite informal to very formal

The goals of the technical review are :

1. To ensure that an early stage the technical concepts are used correctly
2. To access the value of technical concepts and alternatives in the product
3. To have consistency in the use and representation of technical concepts
4. To inform participants about the technical content of the document

3.2.5 Inspection

- An inspection is defined as formal, rigorous, in depth group review designed to identify problems as close to their point of origin as possible. Inspections improve reliability, availability, and maintainability of software product.
- Anything readable that is produced during the software development can be inspected. Inspections can be combined with structured, systematic testing to provide a powerful tool for creating defect-free programs.
- Inspection activity follows a specified process and participants play well-defined roles. An inspection team consists of three to eight members who plays roles of moderator, author, reader, recorder and inspector.
- **For example,** designer can acts as inspector during code inspections while a quality assurance representative can act as standard enforcer.

Stages in the inspections process

1. **Planning :** Inspection is planned by moderator.
2. **Overview meeting :** Author describes background of work product.

3. **Preparation :** Each inspector examines work product to identify possible defects.
4. **Inspection meeting :** During this meeting, reader reads through work product, part by part and inspectors points out the defects for every part.
5. **Rework :** Author makes changes to work product according to action plans from the inspection meeting.
6. **Follow-up :** Changes made by author are checked to make sure that everything is correct.

3.3 DYNAMIC TECHNIQUES

GQ. Explain Dynamic Techniques.

- **Dynamic Testing** is a software testing method used to test the dynamic behaviour of software code.
- The main purpose of dynamic testing is to test software behaviour with dynamic variables or variables which are not constant and finding weak areas in software runtime environment.
- The code must be executed in order to test the dynamic behaviour.
- We all know that Testing is verification and validation, and it takes 2 Vs to make testing complete.
- Out of the 2 Vs, Verification is called a Static testing and the other "V", Validation is known as dynamic testing.

☞ Dynamic Testing Example

- Let's understand How to do Dynamic Testing with an example:
- Suppose we are testing a Login Page where we have two fields say "Username" and "Password" and the Username is restricted to Alphanumeric.
- When the user enters Username as "xyz", the system accepts the same. Whereas when the user enters as xyz @ 123 then the application throws an error message. This result shows that the code is acting dynamically **based on the user input.**
- Dynamic testing is when you are working with the actual system by providing an input and comparing the actual behaviour of the application to the expected behaviour. In other words, working with the system with the intent of finding errors.
- So based on the above statements we can say or conclude that dynamic testing is a process of validating software applications as an end user under different environments to build the right software.

GQ. What does dynamic testing do ?

- The main aim of the Dynamic tests is to ensure that software works properly during and after the installation of the software ensuring a stable application without any major flaws (this statement is made because no software is error free, testing only can show presence of defects and not absence)
- The main purpose of the dynamic test is to ensure consistency to the software; lets discuss this with an example.
- In a Banking Application, we find different screens like My Accounts Section, Funds Transfer, Bill Pay, etc.. All these screens contain amount field which accepts some characters.
- Let's say My Accounts field displays amount as **25,000** and Funds Transfer as **$25,000** and Bill pay screen as **$25000** though the amount is the same, the way amount is displayed is not the same hence making the software no consistent.
- Consistency is not only limited to the functionality it also refers to different standards like performance, usability, compatibity etc, hence it becomes very important to perform Dynamic Testing.

☞ Types of Dynamic Testing

- Dynamic Testing is classified into two categories
  1. White Box Testing
  2. Black Box Testing

- The below pictorial representation gives us an idea about types of Dynamic Testing, Levels of Testing, etc.

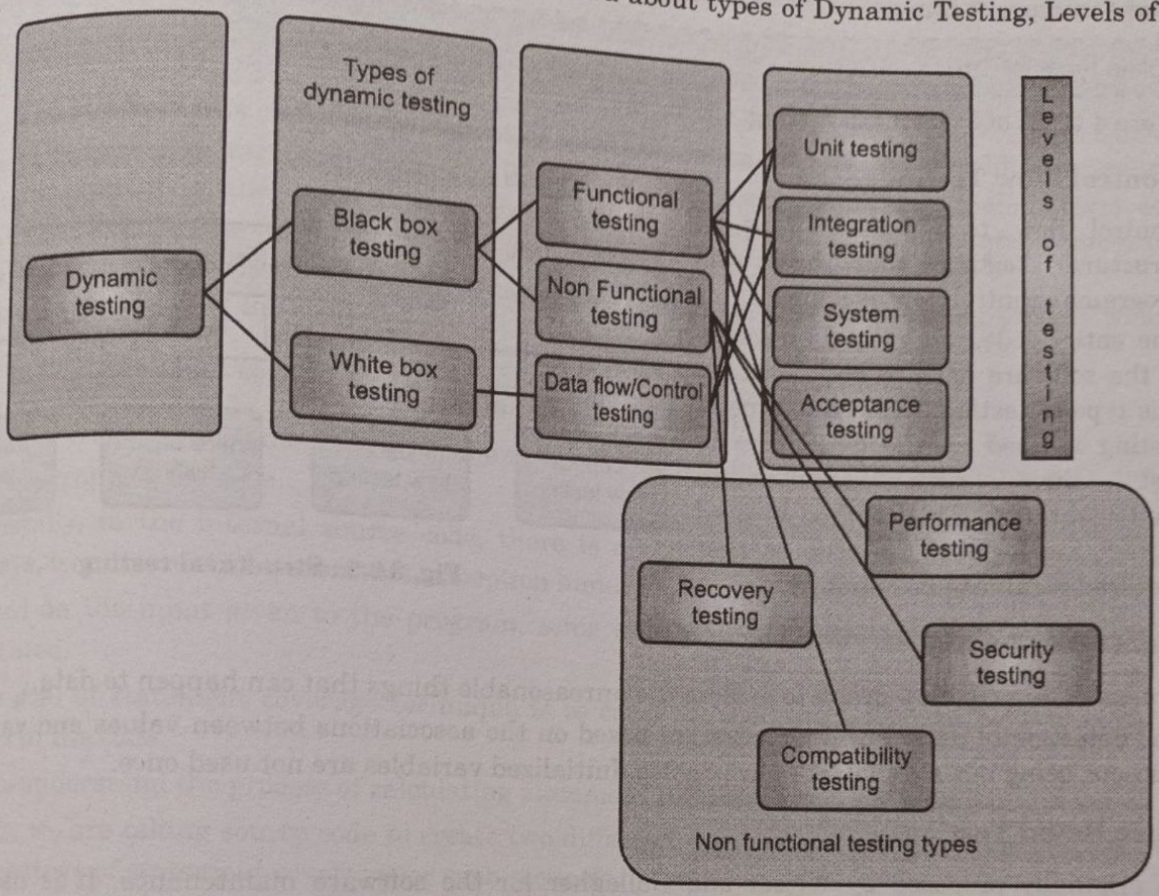


Fig. 3.3.1 : Dynamic Testing

☞ **Let us discuss briefly each type of testing and its intended purpose**

1. **White Box Testing** White Box Testing is a software testing method in which the internal structure/ design is known to the tester. The main aim of White Box testing is to check on how System is performing based on the code. It is mainly performed by the Developers or White Box Testers who has knowledge on the programming.

2. **Black Box Testing** Black Box Testing is a method of testing in which the internal structure/ code/design is **NOT** known to the tester. The main aim of this testing to verify the functionality of the system under test and this type of testing requires to execute the complete test suite and is mainly performed by the Testers, and there is no need of any programming knowledge.

3.3.1 Structural Techniques

GQ. Explain structural Techniques with Example.

- **Structural testing** is a type of software testing which uses the internal design of the software for testing or in other words the software testing which is performed by the team which knows the development phase of the software, is known as structural testing.
- Structural testing is basically related to the internal design and implementation of the software i.e. it involves the development team members in the testing team. It basically tests different aspects of the software according to its types. Structural testing is just the opposite of behavioral testing.

☞ Types of Structural Testing

GQ. Explain types of Structural Testing.

There are 4 types of Structural Testing: (Refer Fig. 3.3.2)

1. Control Flow Testing

- Control flow testing is a type of structural testing that uses the programs's control flow as a model.
- The entire code, design and structure of the software have to be known for this type of testing. Often this type of testing is used by the developers to test their own code and implementation. This method is used to test the logic of the code so that required result can be obtained.

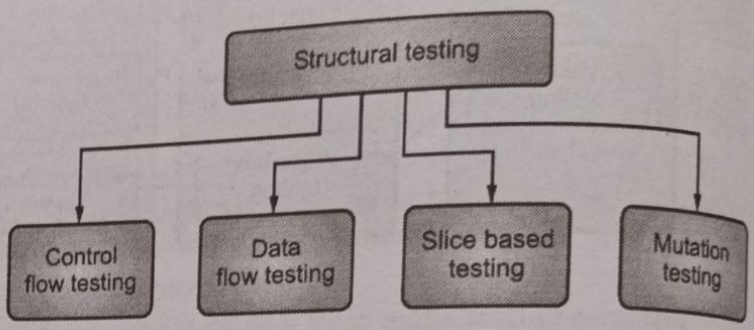


Fig. 3.3.2 : Structural testing

2. Data Flow Testing

- It uses the control flow graph to explore the unreasonable things that can happen to data.
- The detection of data flow anomalies are based on the associations between values and variables. Without being initialized usage of variables. Initialized variables are not used once.

3. Slice Based Testing

It was originally proposed by Weiser and Gallagher for the software maintenance. It is useful for software debugging, software maintenance, program understanding and quantification of functional cohesion. It divides the program into different slices and tests that slice which can majorly affect the entire software.

4. Mutation Testing

- Mutation Testing is a type of Software Testing that is performed to design new software tests and also evaluate the quality of already existing software tests.
- Mutation testing is related to modification a program in small ways. It focuses to help the tester develop effective tests or locate weaknesses in the test data used for the program.

☞ Advantages of Structural Testing

(1) It provides thorough testing of the software.
(2) It helps in finding out defects at an early stage.
(3) It helps in elimination of dead code.
(4) It is not time consuming as it is mostly automated.

☞ Disadvantages of Structural Testing

(1) It requires knowledge of the code to perform test.
(2) It requires training in the tool used for testing.
(3) Sometimes it is expensive.

☞ Structural Testing Tools

- JBehave
- Cucumber
- Junit
- Cfix

3.3.2 Statement Coverage Testing

GQ. Explain statement Coverage Testing with source code structure.

- Statement coverage is one of the widely used software testing. It comes under white box testing.
- Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code out of total statements present in the source code.
- Statement coverage derives scenario of test cases under the white box testing process which is based upon the structure of the code.

$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

- In white box testing, concentration of the tester is on the working of internal source code and flow chart or flow graph of the code.
- Generally, in the internal source code, there is a wide variety of elements like operators, methods, arrays, looping, control statements, exception handlers, etc.
- Based on the input given to the program, some code statements are executed and some may not be executed.
- The goal of statement coverage technique is to cover all the possible executing statements and path lines in the code.
- Let's understand the process of calculating statement coverage by an example:
- Here, we are taking source code to create two different scenarios according to input values to check the percentage of statement coverage for each scenario.

Source Code Structure

- Take input of two values like a = 0 and b = 1.
- Find the sum of these two values.
- If the sum is greater than 0, then print "This is the positive result."
- If the sum is less than 0, then print "This is the negative result."

```
input (int a, int b)
{
Function to print sum of these integer values (sum = a+b)
If (sum>0)
{
Print (This is positive result)
} else
{
Print (This is negative result)
}
}
```

- So, this is the basic structure of the program, and that is the task it is going to do.
- Now, let's see the two different scenarios and calculation of the percentage of Statement Coverage for given source code.

☞ **Scenario 1 : If a = 5, b = 4**

```
print (int a. int b) {
int sum = a+b;
if (sum>0)
print ("This is a positive result")
else
print ("This is negative result")
}
```

- In scenario 1, we can see the value of sum will be 9 that is greater than 0 and as per the condition result will be **"This is a positive result."** The statements highlighted in yellow color are executed statements of this scenario.
- To calculate statement coverage of the first scenario, take the total number of statements that is 7 and the number of used statements that is 5.

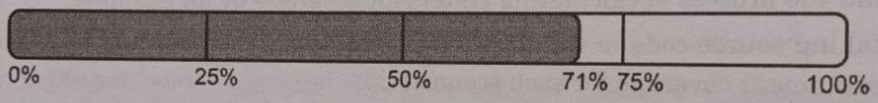
Total number of statements = 7

Number of executed statements = 5

$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

Statement coverage = 5/7*100

= 500/7 = 71 %



Likewise, in scenario 2

☞ **Scenario 2 : If A = -2, B = -7**

```
print (int a, int b) {
int sum = a+b;
if (sum>0)
print ("This is a positive result")
else
print ("This is negative result")
}
```

- In scenario 2, we can see the value of sum will be -9 that is less than 0 and as per the condition; result will be **"This is a negative result."** The statements highlighted in yellow colour are executed statements of this scenario.
- To calculate statement coverage of the first scenario, take the total number of statements that is 7 and the number of used statements that is 6.

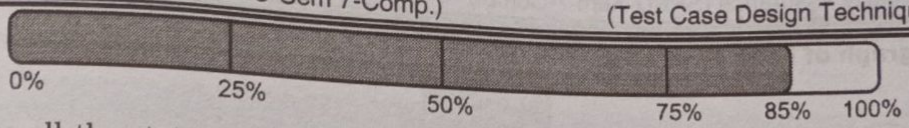
Total number of statements = 7

Number of executed statements = 6

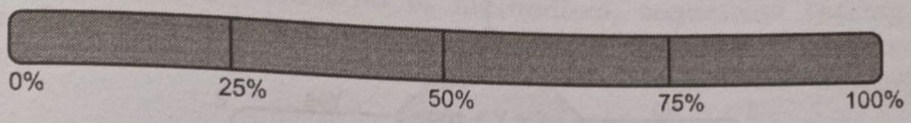
$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

Statement coverage = 6/7*100

= 600/7 = 85 %



- But, we can see all the statements are covered in both scenario and we can consider that the overall statement coverage is 100%.



- So, the statement coverage technique covers dead code, unused code, and branches.

3.3.3 Branch Coverage Testing

GQ. Explain Branch coverage testing with example.

- Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.
- Branch coverage technique is a white box testing technique that ensures that every branch of each decision point must be executed.
- However, branch coverage technique and decision coverage technique are very similar, but there is a key difference between the two.
- Decision coverage technique covers all branches of each decision point whereas branch testing covers all branches of every decision point of the code.
- In other words, branch coverage follows decision point and branch coverage edges. Many different metrics can be used to find branch coverage and decision coverage, but some of the most basic metrics are: finding the percentage of program and paths of execution during the execution of the program.
- Like decision coverage, it also uses a control flow graph to calculate the number of branches.

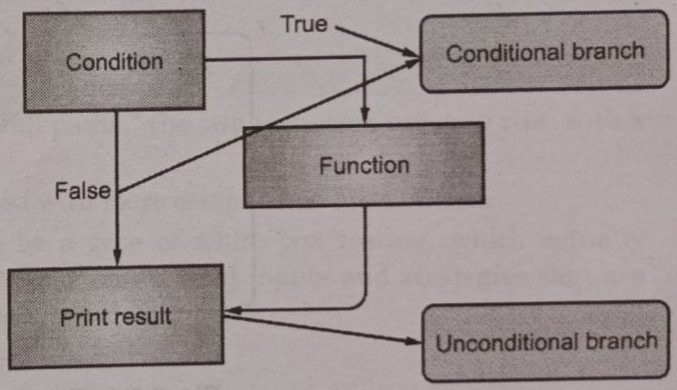


Fig. 3.3.3 : Control Flow Graph

GQ. How to calculate Branch coverage.

- There are several methods to calculate Branch coverage, but path finding is the most common method.
- In this method, the number of paths of executed branches is used to calculate Branch coverage.
- Branch coverage technique can be used as the alternative of decision coverage. Somewhere, it is not defined as an individual technique, but it is distinct from decision coverage and essential to test all branches of the control flow graph.

☞ **Let's understand it with an example :**

1. Read X
2. Read Y
3. IF X + Y > 100 THEN
4. Print "Large"
5. ENDIF
6. If X + Y<100 THEN
7. Print "Small"
8. ENDIF

This is the basic code structure where we took two variables X and Y and two conditions. If the first condition is true, then print "Large" and if it is false, then go to the next condition. If the second condition is true, then print "Small."

☞ **Control flow graph of code structure**

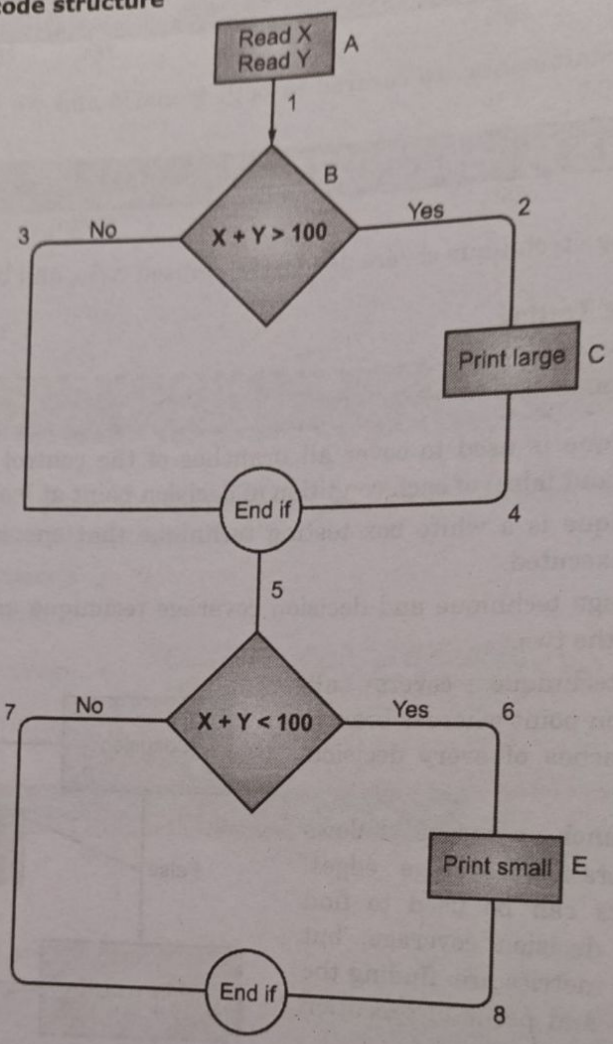


Fig. 3.3.4 : Control flow graph Structure

- In the above diagram, control flow graph of code is depicted. In the first case traversing through "Yes" decision, the path is **A1-B2-C4-D6-E8**, and the number of covered edges is 1, 2, 4, 5, 6 and 8 but edges 3 and 7 are not covered in this path.
- To cover these edges, we have to traverse through "No" decision. In the case of "No" decision the path is A1-B3-5-D7, and the number of covered edges is 3 and 7. So by traveling through these two paths, all branches have covered.

Path 1 - A1-B2-C4-D6-E8

Path 2 - A1-B3-5-D7

Branch Coverage (BC) = Number of paths

= 2

Case	Covered Branches	Path	Branch coverage
Yes	1, 2, 4, 5, 6, 8	A1-B2-C4-D6-E8	
No	3,7	A1-B3-5-D7	

3.3.4 Path Coverage Testing

GQ. What is path coverage testing technique ?

- Path coverage testing is **a specific kind of methodical, sequential testing in which each individual line of code is assessed**. As a type of software testing, path coverage testing is in the category of technical test methods, rather than being part of an overarching strategy or "philosophy" of code.
- It is labor-intensive and is often reserved for specific vital sections of code.
- Techopedia Explains Path Coverage Testing
- The way that path coverage testing works is that the testers must look at each individual line of code that plays a role in a module and, for complete coverage; the testers must look at each possible scenario, so that all lines of code are covered.
- In a very basic example, consider a code function that takes in a variable "x" and returns one of two results : if x is greater than 5, the program will return the result "A" and if x is less than or equal to 5, the program will return the result "B."
- The code for the program would look something like this :

```
input x
if x > 5 then
return A
else return B
```

- In order for path coverage testing to effectively "cover all paths," the two test cases must be run, with x greater than 5 and x less than or equal to 5.
- Obviously, this method becomes much more complicated with more complex modules of code.
- Experts generally consider path coverage testing to be a type of white box testing, which actually inspects the internal code of a program, rather just relying on external inputs and strategies that are considered black box testing, which do not consider internal code.

3.3.5 Conditional Coverage Testing

GQ. What is condition coverage testing ?

- **Condition coverage testing** is a type of white-box testing that tests all the conditional expressions in a program for all possible outcomes of the conditions. It is also called *predicate coverage*.
- **Condition coverage vs. branch coverage** In *branch coverage*, all conditions must be executed at least once. On the other hand, in *condition coverage*, all possible outcomes of all conditions must be tested at least once.
- For example, consider the code snippet below :

```
int a = 10;
if (a > 0){
cout<<"a is positive";
}
```

- *Branch coverage* requires that the condition a > 0 is executed at least once.
- *Condition coverage* requires that both the outcomes a > 0 = True and a > 0 = False of the condition a > 0 are executed at least once.

Examples

Example 1 : Consider the code snippet below, which will be used to conduct *condition coverage testing* :

```
int num1 = 0;
if(num1>0){
cout<<"valid input";
}else{
cout<<"invalid input";
}
```

Condition coverage testing

The condition coverage testing of the code above will be as follows :

Test case number	num1>0	Final output
1	True	True
2	False	False

Example 2 : Consider the code snippet below, which will be used to conduct *condition coverage testing*:

```
int num1 = 0;
int num2 = 0;
if((num1>0 || num2<10)){
cout<<"valid input";
}else{
cout<<"invalid input";
}
```

Condition coverage testing

The condition coverage testing of the code above will be as follows :

Test case number	num1>0	num2<10	Final output
1	True	Not required	True
2	False	True	True
3	False	False	False

Example 3 : Consider the code snippet below, which will be used to conduct *condition coverage testing* :

```
int num1 = 0;
int num2 = 0;
if((num1>0) && (num1+num2<15)){
cout<<"valid input";
}else{
cout<<"invalid input";
}
```

Condition coverage testing

The condition coverage testing of the code above will be as follows :

test case number	Num 1 > 0	num1 + num2 < 15	Final output
1	True	True	True
2	True	False	False
3	False	Not required	False

Example 4 : Consider the code snippet below, which will be used to conduct *condition coverage testing* :

```
int num1 = 0;
int num2 = 0;
if((num1>0 || num2<10) && (num1+num2<15)){
cout<<"valid input";
}else{
cout<<"invalid input";
}
```

Condition coverage testing

The condition coverage testing of the code above will be as follows :

Test case number	Num 1 > 0	Num 2 < 10	num1 + num 2 < 15	Final output
1	True	don't care	True	True
2	True	don't care	False	False
3	False	True	True	True
4	False	True	False	False
5	False	False	Not required	False

3.3.6 Loop Coverage Testing

- Loop Testing is a kind of software testing that focuses exclusively on the correctness of loop constructions. It is a component of Control Structure Testing (path testing, data validation testing, condition testing).
- Loop testing is an example of white-box testing. This approach is used to test software loops.

Types of Loop Tests

The following are some examples of loop tests –

(1) Simple loop (2) Nested loop (3) Concatenated loop (4) Unstructured loop

GQ. Why Loop Testing ?

Following are some of the reasons why loop testing is performed –

- Testing can help to resolve loop recurrence concerns.
- Loop testing can indicate constraints in efficiency and operations.
- The loop's uninitialized variables can be identified by testing loops.
- It aids in the identification of loop initiation issues.

GQ. How to do Loop Testing – Complete Methodology ?

It must be tested at three separate stages within the testing loop –

- When the loop is activated.
- When the loop is executed.
- When the loop is terminated.

The following is the testing technique for all of these loops –

☞ **Simple Loop**

The following is how a simple loop is tested –

- Ignore the entire loop.
- Make a single pass across the loop.
- Make a number of passes through the loop where a<b, n is the maximum limit of passes.
- Make b, b – 1; b + 1 passes through the loop, where "b" is the highest amount of passes through the loop allowed.

☞ **Nested Loop**

You must perform the following steps to create a nested loop –

- Adjust all the other loops to their smallest value and begin with the innermost loop.
- Initiate a simple loop test on the innermost loop and keep the outside loops at their smallest iteration parameter value.
- Conduct the test for the following loop and make your way outwards.
- Keep testing till the outermost loop is reached.

☞ **Concatenated Loops**

- If two loops in a chained loop are free of one other, they are checked as simple loops; otherwise, they are tested as nested loops.
- However, if the loop counter for one loop is utilized as the starting value for the others, the loops are no longer considered separate.

☞ **Unstructured Loops**

For unstructured loops, the architecture must be restructured to represent the use of structured programming techniques.

☞ **Limitation in Loop testing**

- Loop issues are especially common in low-level applications.
- The flaws discovered during loop testing are not significant.
- Numerous defects may be identified by the operating system, resulting in storage boundary breaches, identifiable pointer failures, and so on.

☞ **Summary**

- Loop testing is a type of White Box testing in software engineering. This approach is used to evaluate loops in the program.
- Loop testing can indicate constraints in functionality and operations.
- Loop issues are especially common in low-level applications.

3.4 BLACK BOX TECHNIQUES

- In order to systematically test a set of functions, it is necessary to design test cases.
- Testers can create test cases from the requirement specification document using the following Black Box Testing techniques:

GQ. Explain Black Box Techniques.

3.4.1 Boundary Value Analysis

- The name itself defines that in this technique, we focus on the values at boundaries as it is found that many applications have a high amount of issues on the boundaries.
- Boundary refers to values near the limit where the behavior of the system changes.
- In boundary value analysis, both valid and invalid inputs are being tested to verify the issues.

For Example : (Refer Fig. 3.4.1)

If we want to test a field where values from 1 to 100 should be accepted, then we choose the boundary values : 1 - 1, 1, 1 + 1, 100 – 1, 100, and 100 + 1. Instead of using all the values from 1 to 100, we just use 0, 1, 2, 99, 100, and 101.

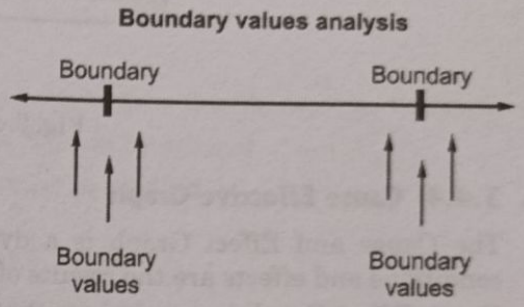


Fig. 3.4.1 : Boundary Value Analysis

3.4.2 Equivalence Class Partition

- This technique is also known as Equivalence Class Partitioning (ECP). In this technique, input values to the system or application are divided into different classes or groups based on its similarity in the outcome.
- Hence, instead of using each and every input value, we can now use any one value from the group/class to test the outcome. This way, we can maintain test coverage while we can reduce the amount of rework and most importantly the time spent.

For Example :

As present in the above image, the "AGE" text field accepts only numbers from 18 to 60. There will be three sets of classes or groups.

Two invalid classes will be :

(a) Less than or equal to 17.

(b) Greater than or equal to 61.

- A valid class will be anything between 18 and 60.
- We have thus reduced the test cases to only 3 test cases based on the formed classes thereby covering all the possibilities. So, testing with any one value from each set of the class is sufficient to test the above scenario.

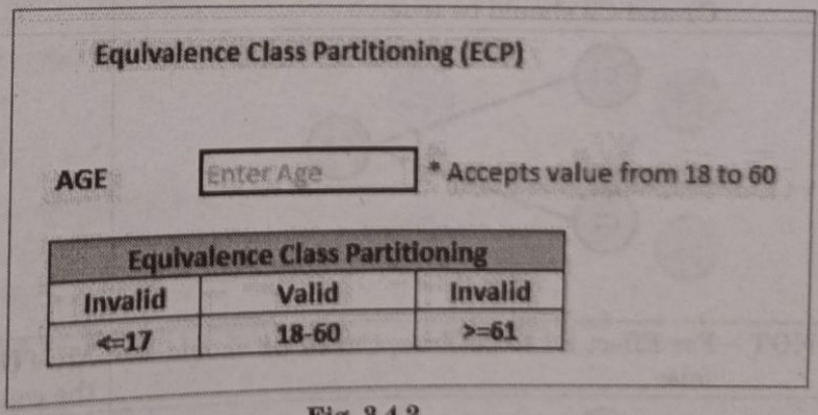


Fig. 3.4.2

3.4.3 State Transition Technique

- State Transition Testing is a technique that is used to test the different states of the system under test. The state of the system changes depending upon the conditions or events. The events trigger states which become scenarios and a tester needs to test them.
- A systematic state transition diagram gives a clear view of the state changes but it is effective for simpler applications. More complex projects may lead to more complex transition diagrams thereby making it less effective.

☞ **For Example :**

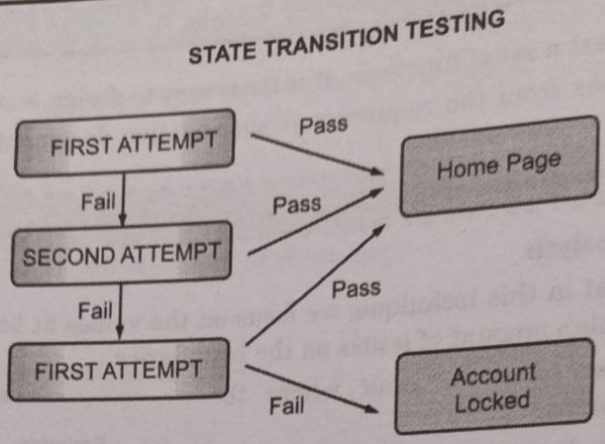


Fig. 3.4.3 : State Transition Testing

3.4.4 Cause Effective Graph

- The Cause and Effect Graph is a dynamic test case writing technique. Here causes are the input conditions and effects are the results of those input conditions.
- Cause-Effect Graph is a technique that starts with a set of requirements and determines the minimum possible test cases for maximum test coverage which reduces test execution time and cost.
- The goal is to reduce the total number of test cases, still achieving the desired application quality by covering the necessary test cases for maximum coverage.
- But at the same time obviously, there are some downsides to using this test case writing technique. It takes time to model all your requirements into this Cause-Effect Graph before writing test cases.
- The Cause-Effect Graph technique restates the requirements specification in terms of the logical relationship between the input and output conditions. Since it is logical, it is obvious to use Boolean operators like AND, OR and NOT.

☞ **Notations Used :**

AND - For effect E1 to be true, both the causes C1 and C2 should be true

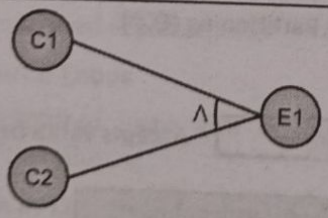


Fig. 3.4.4

OR - For effect E1 to be true, either of causes C1 OR C2 should be true

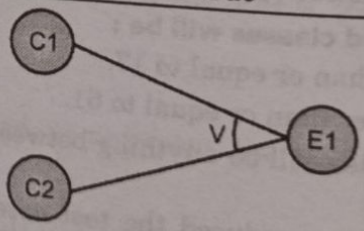


Fig. 3.4.5

NOT – For Effect E1 to be True, Cause C1 should be false

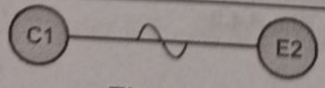


Fig. 3.4.6

MUTUALLY EXCLUSIVE – When only one of the cause will hold true

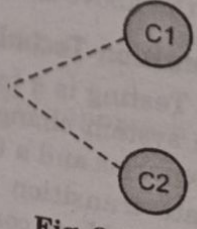


Fig. 3.4.7

☞ **Now let's try to implement this technique with some examples :**

- Draw a Cause and Effect graph based on a requirement/situation.
- Cause and Effect Graph is given, draw a Decision table based on it to draw the test case.
- Let's see both of them one by one.

Example : Draw A Cause And Effect Graph According To Situation

Situation

The "Print message" is software that reads two characters and, depending on their values, messages are printed.

- The first character must be an "A" or a "B".
- The second character must be a digit.
- If the first character is an "A" or "B" and the second character is a digit, then the file must be updated.
- If the first character is incorrect (not an "A" or "B"), the message X must be printed.
- If the second character is incorrect (not a digit), the message Y must be printed.

☑ **Solution :**

The Causes of this situation are :

C1 – First character is A
C2 – First character is B
C3 – the Second character is a digit

The Effects (results) for this situation are :

E1 – Update the file
E2 – Print message "X"
E3 – Print message "Y"

☞ **ET'S START !!**

First, draw the Causes and Effects as shown in Fig. 3.4.8.

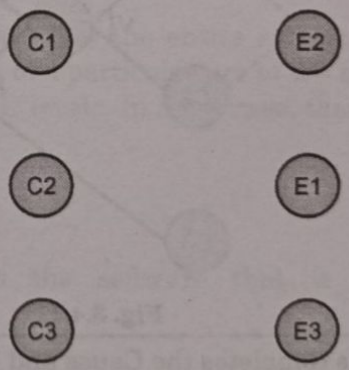


Fig. 3.4.8

Key - Always go from Effect to Cause (left to right). That means, to get effect "E", what causes should be true.

☞ **In this example, let's start with Effect E1.**

Effect E1 is for updating the file. The file is updated when

- The first character is "A" and the second character is a digit
- The first character is "B" and the second character is a digit
- The first character can either be "A" or "B" and cannot be both.

☞ **Now let's put these 3 points in symbolic form :**

For E1 to be true – the following are the causes:

- C1 and C3 should be true
- C2 and C3 should be true
- C1 and C2 cannot be true together. This means C1 and C2 are mutually exclusive.

☞ **Now let's draw this :**

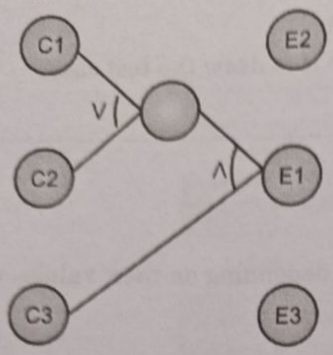


Fig. 3.4.9

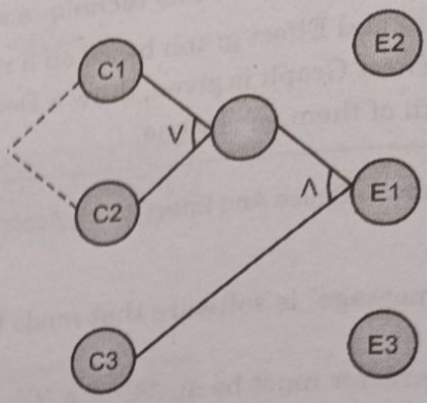


Fig. 3.4.10

So as per shown in Fig. 3.4.9, for E1 to be true the condition is $(C1 \vee C2) \wedge C3$

- The circle in the middle is just an interpretation of the middle point to make the graph less messy.
- There is a third condition where C1 and C2 are mutually exclusive. So the final graph for effect E1 to be true is shown in Fig. 3.4.10.

☞ **Let's move to Effect E2**	☞ **For Effect E3.**
• E2 states print message "X". Message X will be printed when the First character is neither A nor B. • This means Effect E2 will hold true when either C1 OR C2 is invalid. So the graph for Effect E2 is shown in Fig. 3.4.11.	• E3 states print message "Y". Message Y will be printed when the Second character is incorrect. • This means Effect E3 will hold true when C3 is invalid. So the graph for Effect E3 is shown in Fig. 3.4.12.

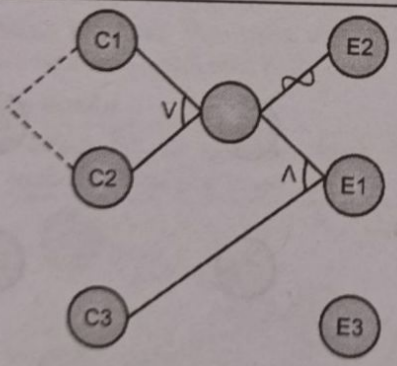


Fig. 3.4.11

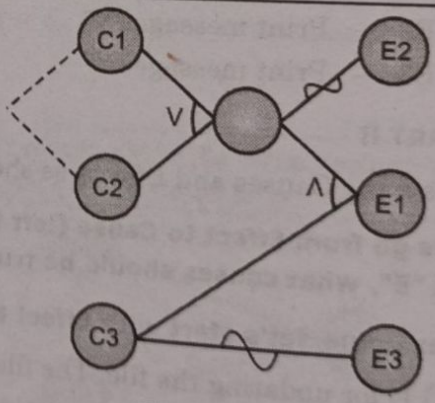


Fig. 3.4.12

- This completes the Cause and Effect graph for the above situation.

3.4.5 Decision Table

GQ. Explain Decision Table with example.

- As the name itself suggests, wherever there are logical relationships like :

```
If
{
   (Condition = True)
   then action1 ;
}
else action2; /*(condition = False)*/
```

- Then a tester will identify two outputs (action1 and action2) for two conditions (True and False). So based on the probable scenarios a Decision table is carved to prepare a set of test cases.

For Example :

- Take an example of XYZ bank that provides an interest rate for the Male senior citizen as 10% and 9% for the rest of the people.

Decision Table / Cause-Effect				
Decision Table	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**
Conditions				
C1 – Male	**F**	**F**	**T**	**T**
C2 – Senior Citizen	**F**	**T**	**F**	**T**
Actions				
A1 – Interest Rate 10%				**X**
A2 – Interest Rate 9%	**X**	**X**	**X**	

Table 3.4.1 : Decision Table / Cause Effect

- In this example condition, C1 has two values as true and false, C2 also has two values as true and false.
- The total number of possible combinations would then be four. This way we can derive test cases using a decision table.

3.4.6 Use Case Testing

GQ. Explain Use case Testing.

What Is A Use Case ?

- Use case testing is a technique that helps to identify test cases that cover the entire system, on a transaction by transaction basis, from start to finish. It is a description of a particular use of the system by a user. It is used widely in developing tests or systems for acceptable levels. In a use case, there will be a set of actions for the user to complete.
- The actions can be :
  - Withdraw funds,
  - Balance transfer, and/or
  - Balance enquiry,
  - Other actions relating to the software that is being developed.

Use Case Example

- When developing use cases, a test case table is usually developed. There will be a success scenario as well as the steps that the user should complete.
- Examples of the steps can be :
  - Insert card,
  - Enters a PIN,
  - Allows access to the account.
  - Validates card and asks for a PIN,
  - Validates a Pin, and
- Following this, there will also be a list of extensions within the table. It could happen, for example, that upon validating the card, the system determines that something is incorrect.

- The extensions can be listed as follows :
  2a) Card not valid (Display message and reject card),
  3a) Pin not valid (Display message and ask for re-try – twice), and
  4a) Pin invalid 3 times (eat card and exit).
- Many times, software testers and developers refer to users as 'actors'. Use cases are associated with the following :
  - Actors (human users, external hardware or other components or systems), and
  - Subjects (the component or system to which the use case is applied).
- Each use case specifies some behaviour that a subject can perform in collaboration with one or more actors.
- A use case specifies a type of behaviour that a subject can perform in collaboration with one or more actors, and it can be described by interactions and activities as well as preconditions, post conditions and natural language where appropriate.
- To start with, let's understand **'What is Use Case?'** and later we will discuss **'What is Use Case Testing ?'**.
- A use case is a tool for defining the required user interaction. If you are trying to create a new application or make changes to an existing application, several discussions are made. One of the critical discussions you have to make is how you will represent the requirement for the software solution.
- Business experts and developers must have a mutual understanding about the requirement, as it's very difficult to attain.
- Any standard method for structuring the communication between them will really be a boon. It will, in turn, reduce the miscommunications and here is the place where Use case comes into the picture.

3.5 EXPERIENCED BASED TECHNIQUES

GQ. Explain Experienced Based Techniques.

- Black Box techniques are used to derive test cases from the specification available. White Box techniques supplement the Black Box techniques and use the code to derive test cases and increase the test coverage.
- How do we test when there is insufficient/no documentation and also a limited time frame ? What do we use to derive test cases and test results in such a scenario ?
- Experience-based techniques come in handy in such situations. They use the tester's experience to understand the most important areas of a system i.e. areas most used by the customer and areas that are most likely to fail. They tap into the tester's experience of defects found in the past when testing similar systems.
- Even when specifications are available, it helps adding tests from past experience.

GQ. What are Different types of experience-based testing techniques?

There are mainly two techniques under this category

(1) Error Guessing (2) Exploratory Testing

3.5.1 Error Guessing

GQ. Explain Error Guessing with example based testing.

- Error Guessing is a simple technique that takes advantage of a tester's skill, intuition and experience with similar applications to identify special tests that formal Black Box techniques could not identify. For example, pressing the Esc key might have crashed a similar application in the past or pressing the

Back button or Enter key on a webpage, JavaScript errors etc.

- This technique completely depends on the tester's experience. The more experienced the tester is, the more errors he can identify.
- Several testers and/or users can also team up to document a list of possible errors, and this can add a lot of value.
- Another way of error guessing is the creation of defect and failure lists. These lists can use available defect and failure data as a starting point and can then expand by using the testers' and users' experience. This list can be used to design tests and this systematic approach is known as fault attack.

This is a classic example of Experience Based Testing

- In this technique, the tester can use his/her experience about the application behaviour and functionalities to guess the error-prone areas.
- Many defects can be found using error guessing where most of the developers usually make mistakes.

Few common mistakes that developers usually forget to handle

- Divide by zero.
- Accepting the Submit button without any value.
- File upload with less than or more than the limit size.
- Handling null values in text fields.
- File upload without attachment.

3.5.2 Exploratory Testing

GQ. Write a note on Exploratory Testing ?

- This is used when specifications are either missing or inadequate and where there is severe time pressure.
- A test charter of test objectives is first created and based on the test objectives, test cases are designed, executed and logged, all these activities occur concurrently along with learning about the application within a fixed time frame (time-box).
- The test objectives help maintain the focus and maximize testing within the limited time frame.

3.6 LEVELS OF TESTING

- Testing levels are the procedure for finding the missing areas and avoiding overlapping and repetition between the development life cycle stages.
- We have already seen the various phases such as **Requirement collection, designing, coding testing, deployment, and maintenance** of SDLC (Software Development Life Cycle).

3.6.1 Functional Testing

GQ. Write a note on Functional Testing ?

- Functional testing is a type of black-box testing. It does not require any knowledge of code. Functional testing is done to test the functionality of the product i.e. the product functions as expected by the customer.
- Functionality is verified by providing the input data and verifying that the output is as expected as in the requirement document.
- Functionality testing covers the functionality of a product, usability i.e. application or product is easy to use, etc.

☞ **Types of Functional Testing**

1. Unit Testing 2. Integration Testing 3. System Testing 4. Acceptance Testing

☞ **Testing is performed in the below order :**

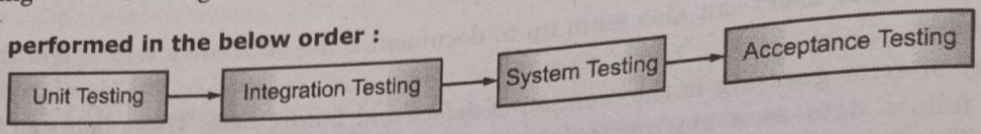


Fig. 3.6.1

3.6.1.1 Unit Testing

GQ. Explain Unit Testing with example.

- Unit testing is the testing of individual components of a product. This testing is done by the developer itself.
- Unit test cases are prepared before the test execution by the developers only. It is very important to execute all the unit test cases as it helps to detect the defect at an early stage.
- Moreover, if the defect is not detected in unit testing, then it could lead to Major and Critical issues at the later stage which indeed would cost high to fix the same and will also take more time than the estimated time.
- While doing unit testing if the developer integrates two components to test to save time and the other component is not yet ready, then the developer uses stubs or drivers to perform his testing. If a defect occurs on the integrated system, then it becomes difficult to find the module, because of which the issue has occurred.
- Thus, testing a component individually is very important.
- Stubs and drivers are basically used when one component calls the function from another component that is not yet developed, thus to create the test environment for testing of the unit, stubs and drivers are used by the developers.

☞ **Stubs**

- A stub is a dummy code that has the same Input parameters as the actual module but has a simplified behaviour.
- Stub follows a **top-down integration approach** i.e. stub is used when the sub-modules are not developed.

☞ **Example :**

If there are three modules i.e. module X, module Y, module Z. Module X is developed and it calls a function from module Y & Z but modules Y and Z are under development. In this case, the stub of module Y & Z is created to test module X. (Refer Fig. 3.6.2)

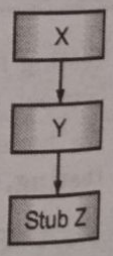


Fig. 3.6.2

☞ **Drivers**

- The driver follows the **bottom-up integration** approach. It is used when the main module is not ready and the other module needs to be tested.
- In that case, a dummy is created for module X so that Module Y & Z can call the function from Module X.

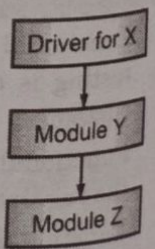


Fig. 3.6.3

3.6.1.2 Integration Testing

GQ. Write a note on Integration Testing with example.

- Integration testing is the testing when two or more modules are integrated or merged to test if the product works fine.
- Modules are integrated in a planned manner as per the integration plan.

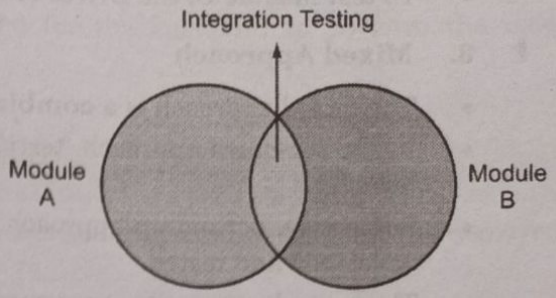


Fig. 3.6.4 : Integration Testing

Example

- The computer which we use has individual components as a Monitor, Screen, Key Board, and Mouse.
- Once the monitor and keyboard are ready, they should be integrated to verify if the output is as expected.

Approaches used in Integration Testing are:

GQ. What are the approaches used in Integration Testing.

1. Top-Down approach 2. Bottom-up approach 3. Mixed approach 4. Big Bang approach

1. Top-down approach

- In Top-down integration testing, the top-level module is developed and tested first. After that immediate sub-modules are integrated with the top-level module and are tested.
- Stubs are required to complete the integration testing with a top-level module in case the module to be integrated is not developed and tested.

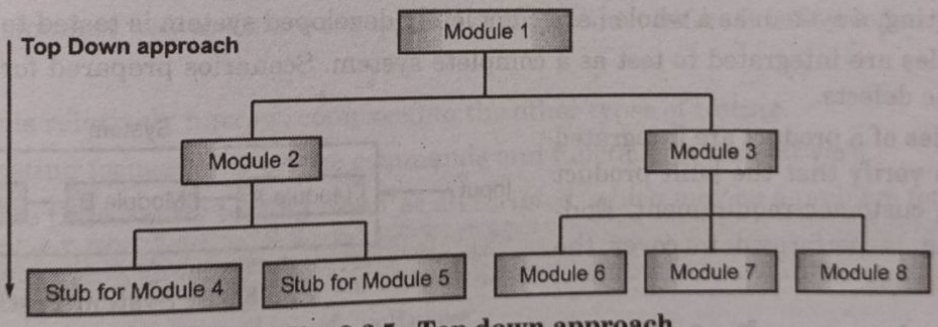


Fig. 3.6.5 : Top down approach

- To test module 2 in integration testing when modules 4 & 5 are not developed, stubs are created to test the same in the top-down approach.

2. Bottom-up Approach

- In the bottom-up approach, sub-modules are developed and tested first and the whole system is tested.

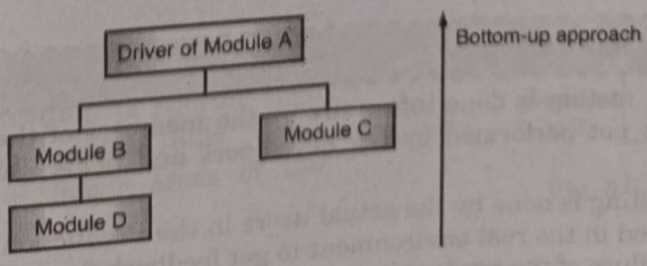


Fig. 3.6.6 : Bottom Up Approach

- To test Module C, the Driver of module A is created so that function can be called.

3. Mixed Approach

- The mixed approach is a **combination of both top-down and bottom-up** approaches.
- In the top-down approach, testing can start only when the top modules are developed and the unit tested.
- Same way, bottom-up approach testing, can start only when the modules at the bottom have been developed and tested.
- The mixed approach overcomes this shortcoming as in the mixed approach, testing can start anytime once the modules are developed.

4. Big Bang Approach

- In the Big bang **approach, all the modules are integrated** into one go and are tested.
- A drawback of this method is that if any error is detected while testing, it would be difficult to find the error-causing module.
- Once the defect cause is detected, it will be fixed but it would cost high & will be time-consuming as the defect found is at a later stage.

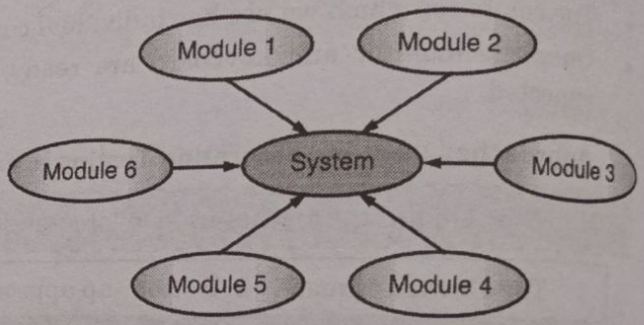


Fig. 3.6.7 : Big Bang Approach

3.6.1.3 System Testing

GQ. Write a note on System Testing.

- In system testing, a system as a whole i.e. a completely developed system is tested to find defects.
- All the modules are integrated to test as a complete system. Scenarios prepared for system testing are run to find the defects.
- All the modules of a product are integrated and tested to verify that the built product is as per the customer requirement. End-to-end testing is performed to cover the complete scenarios.

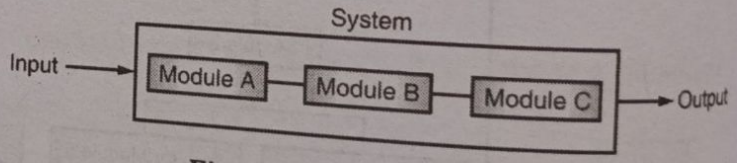


Fig. 3.6.8 : System Testing

3.6.1.4 User Acceptance Testing

- Once the System testing is complete, the Product is ready to be released to the Production environment. But before deployment, the customer's acceptance and approval are required.
- The customer verifies whether the product is all that they wanted or not. For this acceptance testing is done.

Types of Acceptance Testing

GQ. Explain types of Acceptance Testing.

1. **Alpha Testing :** Alpha testing is done internally by the members of the company who developed the product. This testing is not performed by the developers and testers who developed and tested the product.
2. **Beta Testing :** Beta testing is done by the actual users in the Production environment. Beta version of the application is released in the real environment to get feedback from the users which indeed helps to reduce the chances of failure of the product in the production environment.

3. **User Acceptance Testing :** Testing is done at the user end wherein the prepared UAT test cases are executed. Replica of the production environment is created for the customer to perform the user acceptance testing.

3.6.2 Sanity/Smoke Testing

GQ. Explain Sanity / Smoke Testing with example.

Smoke Testing can be understood as a software testing process that determines whether the deployed software build is stable or not.

Some Pros

- It helps to find issues in the early phase of testing.
- Improves the overall quality of the system.
- Very limited number of test cases is required to do the Smoke testing.

Some Cons

- Smoke testing does is not detailed.
- Smoke testing sometimes causes wastage of time if the software build is not stable.
- This type of test is more suitable if you can automate; otherwise, a lot of time is spent manually executing the test cases.

Sanity Testing is a type of software testing that is performed after receiving a software build. The goal is to determine that the proposed functionality works approximately as expected.

Some Pros

- It helps to find related and missing objects.
- It saves lots of time and effort because it is focused on one or few areas of functionality.
- It is the simplest way to assure the quality of the product before it is developed.

Some Cons

- Its scope is relatively narrow, compared to the other types of testing.
- Sanity testing focuses only on the commands and functions of the software.
- Most of the time Sanity testing is not at all scripted, which may be a problem for some testers.

GQ. What is Smoke testing and Sanity testing ?

- Smoke testing in the practice of software development and software testing has become a commonly used technique for the early and continuous detection of software defects.
- There are many ways that smoke testing can be applied in software applications and systems.
- Smoke testing, also known as build verification testing or confidence testing, is shorter than the Sanity test and includes only quick scenarios that point at major areas of the product.

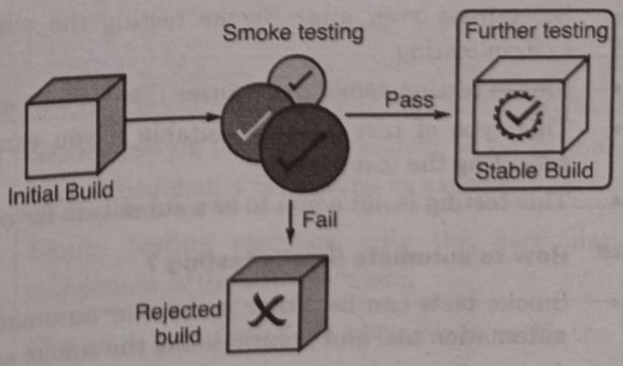


Fig. 3.6.9 : Smoke Testing

- It is performed on initial builds before they are released for extensive testing. Smoke testing is one of the important functional testing types.
- The main idea of this test is to probe these areas in search of smoke that will signal there is a problem hiding below the surface.
- On the other hand, Sanity testing is a type of software test that is performed on a stable build after minor changes to code or functionality. The goal is to determine if the proposed functionality works as expected.

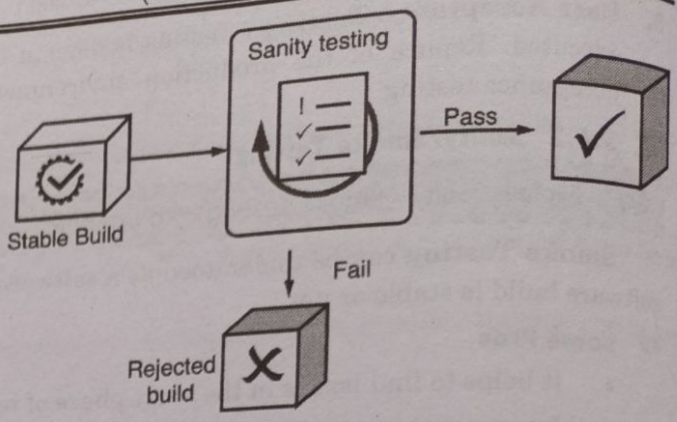


Fig. 3.6.10 : Sanity Testing

- Sanity testing is similar to Regression testing in the fact that you choose specific scenarios that cover all the AUT or application area but shorter and based on the highest risk factors. It is executed at an end in the process of a software development Lifecycle.

☞ **Pros of Smoke testing**

- Smoke tests are very useful when you either don't have time or don't know where to start and need a quick test to show you the level of your product.
- The results of this test help decide if a build is stable enough to continue with further testing.
- It helps to find issues in the early phase of testing.
- It helps to find issues introduced in integration of modules.
- Improves the effectiveness of the QA team. Many of the problems could be discovered with a Smoke test, thus helping to save time and resources.
- Faster troubleshooting of new and regression bugs. This is due to the high coverage, shallow depth nature of Smoke testing suites.
- Very limited number of test cases is required to do the Smoke testing.
- Smoke testing is quite easy to perform as no special efforts of the testing team are required.
- Improves the overall quality of the system.

☞ **Cons of Smoke testing**

- Smoke testing does not cover detailed testing.
- Sometimes even after Smoke testing the whole application, critical issues arise in integration and system testing.
- Smoke testing sometimes causes the wastage of time if the software build is not stable.
- This type of test is more suitable if you can automate; otherwise, a lot of time is spent manually executing the test cases.
- This testing is not equal to or a substitute for complete functional testing.

☞ **How to automate Smoke testing ?**

- Smoke tests can be either manual or automated, but the best way to do a Smoke test is by using an automation tool and programming the smoke suite to run when a new build is created.
- Automation testing is an efficient method that allows to automate most of the testing efforts and there are many tools available to perform this type of test.

- One place where you can find all the tools you need to automate your Smoke test is PractiTest. It is an end-to-end test management tool that gives you control of the entire testing process - from manual testing to automated testing and CI.
- Designed for testers by testers, PractiTest can be customized to your team's ever-changing needs.
- With fast professional and methodological support, you can make the most of your time and release products quickly and successfully to meet your user's needs.
- Some tools that can be used to perform the Smoke Automation test process are:
  - Selenium
  - Robotium
  - Test Complete
  - Appium
  - Cucumber
  - Watir
  - Jenkins (CI tool)
  - Calabash

☞ **Pros of Sanity testing**

- It saves lots of time and effort because it is focused on one or few areas of functionality.
- It is used to verify that a small functionality of the application is still working fine after a minor change.
- Since no documentation is required, these tests can be carried out in a lesser time as compared to other formal tests.
- In case issues are found during Sanity testing, the build is rejected. This saves a lot of time and resources.
- It is the simplest way to assure the quality of the product before it is developed.
- Sanity testing is simple to understand and to carry out and very effective in detecting bugs.
- It can reduce the effort during Smoke and Regression testing.
- It helps to find related and missing objects.

☞ **Cons of Sanity testing**

- Its scope is relatively narrow, compared to the other types of testing.
- Sanity testing focuses only on the commands and functions of the software.
- Most of the time Sanity testing is not at all scripted, which may be a problem for some testers.
- Sanity testing goes nowhere near the design structure level, so it's very difficult for the developers to understand how to fix the issues found during the Sanity testing.

☞ **How to automate Sanity testing ?**

- Sanity tests can be either manual or automated. But beyond that, Sanity tests are more often scripted and automated because they are derived from existing Regression tests.
- Automation saves time and provides faster results for the development team to take action on.
- One place where you can find all the tools you need to automate your Sanity testing is PractiTest.

GQ. What are important differences: Smoke vs Sanity testing ?

Smoke Testing	Sanity Testing
Smoke Testing is performed to ascertain that the critical functionalities of the program are working fine.	Sanity testing is done at random to verify that each functionality is working as expected.
Smoke testing exercises the entire system from end to end.	Sanity testing exercises only the particular component of the entire system.
The main objective of the testing is to verify the stability of the system.	The main objective of the testing is to verify the rationality of the system.
Smoke testing is usually documented and scripted.	Sanity testing is not documented and is unscripted.

Smoke Testing	Sanity Testing
This testing is performed by the developers or testers.	Sanity testing in software testing is usually performed by testers.
It is a well elaborate and planned testing.	This is not a planned test and is done only when there is a shortage of time.
This is a wide and deep testing.	This is a wide and shallow testing.
Smoke testing is a subset of Acceptance testing.	Sanity testing is a subset of Regression Testing.

Similarities between Smoke and Sanity Testing	Explanation
Save time	Smoke and Sanity tests save time by quickly determining whether an application is working properly or not.
Save cost	Due to time and effort savings, costs are reduced.
Integration risk	Integration problems are minimized because end-to-end testing is performed on every build so that functionality-based problems are discovered earlier.
Quality improvement	The main problems are detected and corrected much earlier in the software test cycle, which increases the quality of the software.
Evaluation of progress	Assessing development progress becomes an easier task. Since with each compilation, it is certified that the end-to-end product is working correctly after the addition of new features.

3.6.3 Regression Test

GQ. What is Regression Testing ? Definition, Tools & How to Get Started ?
GQ. What is regression testing ?

- Regression testing is a software testing practice that ensures an application still functions as expected after any code changes, updates, or improvements.
- Regression testing is responsible for the overall stability and functionality of the existing features. Whenever a new modification is added to the code, regression testing is applied to guarantee that after each update, the system stays sustainable under continuous improvements.
- Changes in the code may involve dependencies, defects, or malfunctions. Regression testing targets to mitigate these risks, so that the previously developed and tested code remains operational after new changes.
- Generally, an application goes through multiple tests before the changes are integrated into the main development branch. Regression testing is the final step, as it verifies the product behaviors as a whole.

GQ. When to apply regression testing?

Typically, regression testing is applied under these circumstances:

- A new requirement is added to an existing feature
- A new feature or functionality is added
- The codebase is fixed to solve defects

- The source code is optimized to improve performance
- Patch fixes are added
- Changes in configuration

GQ. Why is regression testing important ?

- Test automation is a necessary element in software development practices. Similarly, automated regression testing is also considered a critical puzzle piece.
- With a rapid regression testing process, product teams can receive more informative feedback and respond instantly.
- Regression testing detects new bugs early in the deployment cycle so that businesses do not have to invest in costs and maintenance efforts to resolve the built-up defects.
- Sometimes a seemingly mild modification might cause a domino effect on the product's key functions.
- That's why developers and testers must not leave any alteration, even the smallest, that goes out of their control scope.

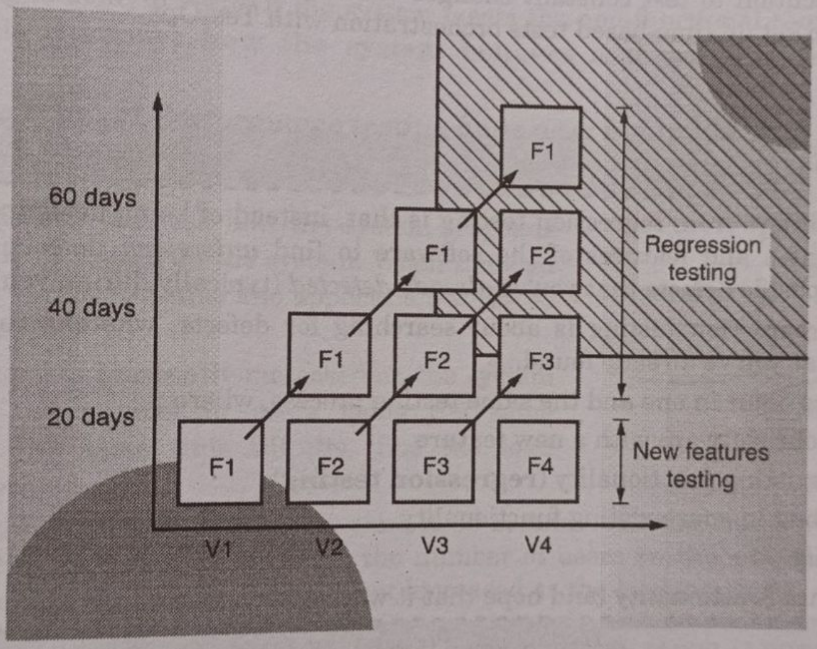


Fig. 3.6.11 : Regression testing

- Functional tests only inspect behaviours of the new features and capabilities, yet dismiss how compatible they are with the existing ones. Therefore, without regression testing, it is more difficult and time-consuming to investigate the root cause and the architecture of the product.
- In other words, if your product undergoes frequent modification, regression testing will be the filter that ensures quality as the product is improved.

GQ. How to perform regression testing ?

Regression testing practices vary among organizations. However, there are a few basic steps :

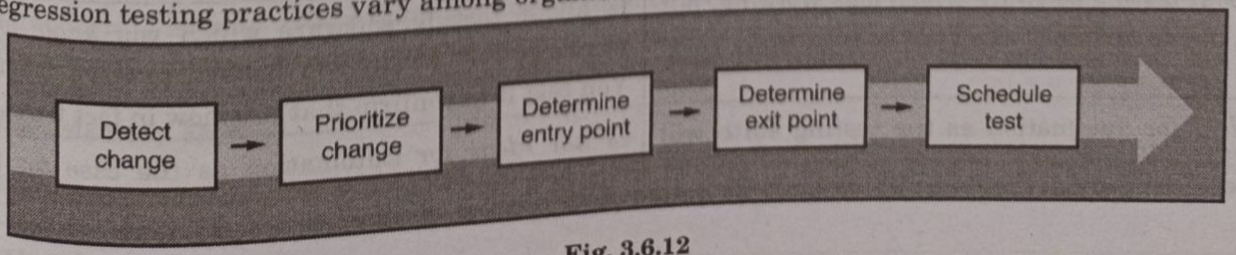


Fig. 3.6.12

Detect Changes in the Source Code

Detect the modification and optimization in the source code; then identify the components or modules that were changed, as well as their impacts on the existing features.

1. **Prioritize Those Changes and Product Requirements**
   Next, prioritize these modifications and product requirements to streamline the testing process with the corresponding test cases and testing tools.
2. **Determine Entry Point and Entry Criteria**
   Ensure whether your application meets the preset eligibility before the regression test execution.
3. **Determine Exit Point**
   Determine an exit or final point for the required eligibility or minimum conditions set in step three.
4. **Schedule Tests**
   Finally, identify all test components and schedule the appropriate time to execute.
   Is time-based execution to test constant changes one of your priorities in testing? If so, learn how to easily manage and set up time-based tests orchestration with TestOps.

3.6.4 Retest

GQ. What is retesting?

- Where retesting differs from regression testing is that, instead of being designed to search through all the previous updates and features of the software to find *unforeseen* defects and bugs, retesting is designed to test specific defects that you've *already detected* (typically during your regression testing).
- In other words, regression testing is about searching for defects, whereas retesting is about fixing specific defects that you've already found.
- They can therefore occur in one and the same testing process, where :
  1. You update your software with a new feature
  2. You test the existing functionality (**regression testing**)
  3. You detect a bug in your existing functionality
  4. You fix the bug
  5. You **retest** that functionality (and hope that it works!)

GQ. What is Regression testing vs. retesting: key differences?

- You could say that regression testing is a *type of* retesting. Retesting essentially means to test something again. And when you are regression testing, you're testing something that you've tested numerous times before.
- But determining what the two have in common might confuse more than it will help. So for the sake of clarity, here's an overview of the key differences.

Regression Testing	Retesting
Involves testing a *general area* of the software.	Involves testing a *specific feature* of the software.
Is about testing software which *was* working, but now, due to updates, *might not* be working.	Is about testing software which you *know was not* working, but which you believe to have been fixed. You test it to confirm that it is now in fact fixed.
Is ideal for automation as the testing suite will grow with time as the software evolves.	*Is not ideal* for automation as the case for testing changes each time.

3.7 NON-FUNCTIONAL TESTING

- Non-functional testing is done to verify the non-functional aspects of a product. **E.g.** Performance of any product comes under non-functional activity.
- Non-functional testing is as important as functional testing is. If it is not done properly, it can lead to major issues in the Product.

Example :

If Performance testing is not done for new functionality added to the application, when the application goes live it becomes too slow to work with and the customer gets affected with the same.

3.7.1 Performance Testing

GQ. Explain performance testing with example.

- Performance testing is done to verify if the system meets the non-functional requirement identified in the SRS document. It verifies **how the system behaves and performs like response time, throughput,** etc.
- **There are various types of Performance testing done as given in Fig. 3.7.1.**

1. **Load Testing**
   - Load testing is done to check how the system responds when the load is increased on the system. When the load is at a peak, it verifies whether the application works as expected or not ?
   - Basically, the load is constantly increased on the system by increasing the number of users using the same application at the same time till the time the load reaches its threshold.

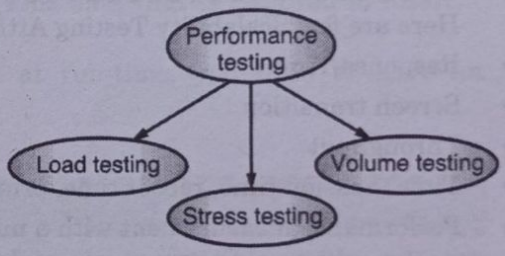


Fig. 3.7.1 : Performance Testing

   - **Example :** If the application's response time is 30 seconds with the number of logged-in users as 1000, then testing is done by increasing the number of users on the site till the number reaches 1000. It verifies that the response time is not increased as the load increases.

2. **Stress Testing**
   - Stress testing is **done beyond the system's capacity** to check at which point it fails. In this, invalid or illegal inputs are used to test so as to stress the capabilities of a product.
   - Stress is increased on the system by increasing the number of users till the time the application breaks.
   - **Example :** If the application's response time is 30 seconds with the number of logged-in users as 1000, then testing is done by increasing the number of users on the site till the number reaches (more than 1000 users) wherein the application breaks down. The number of users is increased beyond the capability of the product to handle it.

3. **Volume Testing**
   Volume testing is done with a huge amount of data **to verify the efficiency & response time of the software** and also to check for any data loss.

3.7.2 Memory Test

If ever there was a piece of embedded software ripe for reuse it is the memory test. This article shows how to test for the most common memory problems with a set of three efficient, portable, public-domain memory test functions.

3.7.3 Scalability Testing

- **Scalability Testing** is a non-functional testing method that measures performance of a system or network when the number of user requests is scaled up or down.
- The purpose of Scalability testing is to ensure that the system can handle projected increase in user traffic, data volume, transaction counts frequency, etc. It tests system ability to meet the growing needs.
- It is also referred to as performance testing, as such; it is focused on the behaviour of the application when deployed to a larger system or tested under excess load.
- In Software Engineering, Scalability Testing is to measure at what point the application stops scaling and identify the reason behind it.

GQ. Why does Scalability Testing ?

- Scalability testing lets you determine how your application scales with increasing workload.
- Determine the user limit for the Web application.
- Determine client-side degradation and end user experience under load.
- Determine server-side robustness and degradation.

GQ. What to test in Scalability Testing ?

Here are few Scalability Testing Attributes :

- Response Time
- Screen transition
- Throughput
- Time (Session time, reboot time, printing time, transaction time, task execution time)
- Performance measurement with a number of users
- Request per seconds, Transaction per seconds, Hits per second
- Performance measurement with a number of users
- Network Usage
- CPU / Memory Usage
- Web Server (request and response per seconds)
- Performance measurement under load

☞ Test Strategy for Scalability testing

Test Strategy for Scalability Testing differs in terms of the type of application is being tested. If an application accesses a database, testing parameters will be testing the size of the database in relation to the number of users and so on.

☞ Prerequisites for Scalability Testing

1. **Load Distribution Capability :** Check whether the load test tool enables the load to be generated from multiple machines and controlled from a central point.
2. **Operating System :** Check what operating systems do the load generation agents and load test master run under
3. **Processor :** Check what type of CPU is required for the virtual user agent and load test master
4. **Memory :** Check how much memory would be enough for the virtual user agent and load test master.

GQ. How to do Scalability Testing ?

1. Define a process that is repeatable for executing scalability tests throughout the application life-cycle
2. Determine the criteria for scalability
3. Shortlist the software tools required to run the load test
4. Set the testing environment and configure the hardware required to execute scalability tests
5. Plan the test scenarios as well as Scalability Tests
6. Create and verify visual script
7. Create and verify the load test scenarios
8. Execute the tests
9. Evaluate the results
10. Generate required reports

Scalability Test Plan

- Before you actually create the tests, develop a detailed test plan. It is an important step to ensure that the test conforms as per the application requirement.
- Following are the attributes for creating a well-defined Test Plan for Scalability Testing.
  1. **Steps for Scripts :** The test script should have a detailed step that determines the exact actions a user would perform.
  2. **Run-Time Data :** The test plan should determine any run-time data that is required to interact with the application
  3. **Data Driven Tests :** If the scripts need varying data at run-time, you need to have an understanding of all the fields that require this data.

3.7.4 Compatibility Testing

GQ. Write a note on : Compatibility Testing.

- Compatibility testing verifies whether the application/Product is compatible with all the hardware/software platforms or not.
- The application should be **compatible with all browsers, operating systems, mobile devices**.
- **Example :** The application might show all the images perfectly in one browser whereas, in another browser, the images are shown distorted. So, it's important to verify the application on all the browsers and operating systems that the customer has asked for in his requirement.

3.7.5 Security Testing

Security testing is one of the important testing methods as security is a crucial aspect of the Product. It is done to verify if the application is secured or not.

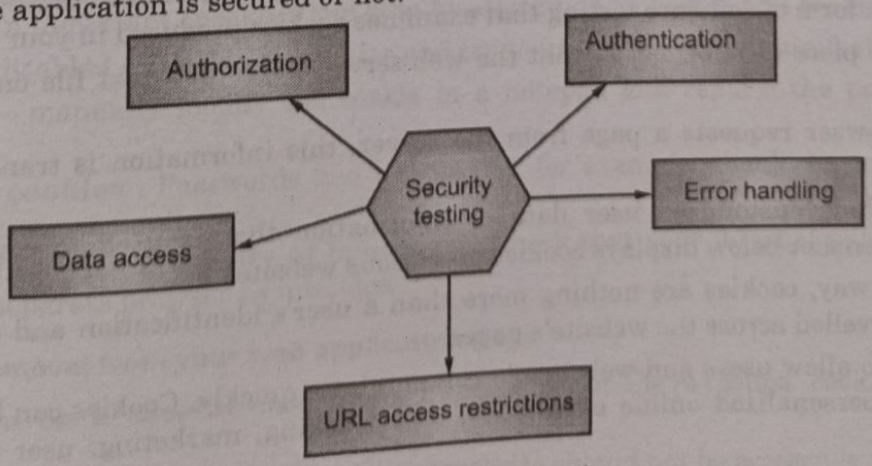


Fig. 3.7.2 : Security Testing

Few examples of what is being tested while performing security testing are :

1. Security testing verifies that only **authorized** users should be able to access data.
2. **Authentication** is verified i.e. the user should be able to log in with his credentials. An incorrect username or password should not allow the user to log in.
3. **Secured and unsecured pages** in the application should be verified.
4. For financial sites, **the session should timeout** after a few minutes if no activity is being performed.

3.7.6 Cookies Testing

GQ. Write a note on Cookies Testing with example.

The Basics of Cookies

- We'll start by discussing what cookies are and how they operate.
- When you understand how cookies operate, it will be much easier for you to comprehend the test cases for testing cookies. How can cookies end up on your computer's hard drive? Also, how can we change our Cookie preferences?

GQ. What Exactly Is a Cookie ?

- A cookie is a little piece of data that a web server stores in a text file on the user are hard disc.
- The web browser then uses this information to obtain information from that machine.

GQ. What is the purpose of cookies ?

- Cookies are simply the user's identification, and they are used to monitor where the user traveled across the website pages. Stateless communication exists between a web browser and a web server.
- As an example
- If you visit the domain http://www.example.com/1.html your web browser will simply ask the example.com web server for page 1.html. If you write the page as http://www.example.com/2.html the next time, a fresh request will be made to the example.com web server for sending the 2.html page, and the webserver has no idea who the prior page 1.html was provided to.
- What if you want to see the history of this user's interactions with the webserver? Somewhere, the user state and interaction between a web browser and a web server must be maintained. This is when the cookie comes into play. Cookies are used to keep user interactions with websites going.

GQ. What exactly is Cookie Testing ?

- Cookie Testing is a form of software testing that examines cookies produced in your web browser.
- A cookie is a little piece of information that the web server stores in a text file on the user's (client's) hard disc.
- Each time the browser requests a page from the server, this information is transmitted back to the server.
- Cookies often include customized user data or information that is used to communicate between websites. The screenshot below displays cookies on various websites.
- To put it another way, cookies are nothing more than a user's identification and are used to monitor where the user travelled across the website's pages.
- A cookie's aim is to allow users and websites to communicate quickly. Cookies can be used to provide a shopping cart, a personalized online experience, user tracking, marketing, user sessions, and other applications.

GQ. What Is the Function of Cookies ?

- The HTTP protocol, which is used to communicate information files on the internet, is used to keep cookies.
- HTTP protocols are classified into two kinds. There are two HTTP protocols: stateless HTTP and stateful HTTP. The stateless HTTP protocol does not maintain track of previously visited web pages. While the Stateful HTTP protocol does preserve some history of prior web browser and web server interactions, cookies employ this protocol to maintain user interactions.
- When a user visits a site or page that uses a cookie, the little code contained inside that HTML page (usually a call to some language script to write the cookie, such as cookies in JAVAScript, PHP, or Perl) creates a text file.
- When a user returns to the same page or site, this cookie is retrieved from the disc and used to identify the user's second visit to that domain. The expiration time is set when the cookie is created. This time is set by the program that will use the cookie.
- In general, two sorts of Cookies are stored on the user's computer.

1. **Session Cookies :** This cookie remains active as long as the browser that set it is open. This session cookie is erased when we exit the browser. The cookie can sometimes be configured to expire after a 20-minute session.
2. **Persistent Cookies :** These are cookies that are stored on the user's computer indefinitely and can last months or years.

GQ. What is the Cookie's Content ?

The cookie is made up of three major components.

- The name of the server from which the cookie was sent
- Lifetime of Cookies
- A monetary value. This is generally a one-of-a-kind number created at random.

GQ. Where do cookies get saved ?

- Any web page program that writes a cookie saves it in a text file on the user's hard drive. The location of the cookies is determined by the browser. Cookies are stored in various routes by different browsers.
- Cookies, for example, maybe seen in the browser settings in Mozilla Firefox. To see it, go to Tools -> Options -> Privacy and then "Remove Individual Cookies."
- Cookies are stored on the location "C:\Documents and Settings\Default User\Cookies" in the Internet Explorer browser.

☞ **How to Put Cookies to the Test**

The following is an important checklist and step-by-step guide on how to test cookies on a website –

- Cookies must be disabled – Disable all cookies and try to utilize the site's main features.
- Cookies tainted – manually modify the cookie in a notepad and replace the parameters with some arbitrary ones.

1. **Encrypting cookies :** Passwords and usernames, for example, should be encrypted before being transferred to our computer.
2. **Cookie testing with a variety of browsers :** Check that your internet page is correctly writing cookies on a separate browser as intended.

☞ **Examining the removal from your web application page**

- **Cookie rejection on a case-by-case basis :** Delete all of the websites' cookies and see how the website behaves.
- **Cookies are accessible :** Cookies created by one website should not be accessible to other websites.

- **Cookies should not be used excessively :** If the program under test is a public website, cookies should not be used excessively.
- **Experimenting with various settings :** Testing should be done thoroughly to ensure that the website works correctly with various cookie settings.
- **Separately categorize cookies :** Cookies should not be classified in the same category as viruses, spam, or malware.

☞ **Specific Test for multi-environment sites**

- A test that is specific Check if the same Cookies are allowed in all environments for multi-environment sites.
- The usage of wildcards in the Cookie path might be the reason (so-called super cookies). If this is a need, certain access difficulties may arise as a result of the alternative encryption key being utilized (for.Net it is a machine key that usually is unique unless specified otherwise).
- These are some of the most important test scenarios to keep in mind while evaluating website cookies.
- You may create many test cases from these test cases by combining them in different ways. If you have an alternative application situation, please share it in the comments section below.

☞ **Cookies can be used in the following ways**

1. **To implement the shopping cart :** Cookies are used to keep an online ordering system running smoothly. Cookies keep track of what the user wishes to purchase. What if a consumer adds certain goods to their shopping cart and then decides not to buy them this time because of whatever reason and shuts the browser window? When the same user returns to the buy page, he will be able to see all of the goods he put in his shopping basket on his previous visit.
2. **Personalized sites :** When a person views a certain website, they are asked which other pages they do not wish to see. User preferences are saved in a cookie, and those pages are not displayed to the user until he is online.
3. **User tracking :** Counting the number of unique visitors who are online at any one moment.
4. **Advertising :** Some businesses use cookies to display advertising on user computers. These adverts are controlled by cookies. When and how should advertisements be shown? What is the user's point of view? What keywords do they look upon the site? Cookies may be used to keep track of all of these things.
5. **User sessions :** Using a user ID and password, cookies may monitor user sessions to a certain domain.

☞ **Drawbacks of Cookies**

- While writing a Cookie is a great way to keep users engaged, if the user has set their browser to warn them before writing any Cookies or has completely disabled cookies, the site containing the Cookie will be completely disabled and unable to perform any operations, resulting in a loss of site traffic. This may be turned off or on in your browser's settings. For Google Chrome, for example, go to Settings -> Advanced -> Content Settings -> Cookies. You may apply a cookie policy to all websites or set it up for specific ones.
- In addition to browser settings, changes in EU and US regulations require developers to notify users that cookies are being used on their websites.
- Compliance with such new rules should be included in test scenarios for specific areas.

1. **Too many Cookies :** If you are writing too many cookies on every page navigation and the user has enabled the option to warn before writing the Cookie, this may turn away users from your site.
2. **Security Concerns :** Personal information about users is sometimes saved in Cookies, and if the Cookie is hacked, the hacker can access your personal information. Even damaged cookies can be read by several websites, posing security risks.
3. **Sensitive Information :** Some websites may write and keep sensitive information about you in cookies, which is not permitted owing to privacy concerns. This should be sufficient to understand what Cookies are.

3.7.7 Session Testing

GQ. Write a note on session testing.

- **Session-based testing** is a software test method that aims to combine accountability and exploratory testing to provide rapid defect discovery, creative on-the-fly test design, management control and metrics reporting.
- The method can also be used in conjunction with scenario testing. Session-based testing was developed in 2000 by Jonathan and James Bach.
- Session-based testing can be used to introduce measurement and control to an immature test process and can form a foundation for significant improvements in productivity and error detection.
- Session-based testing can offer benefits when formal requirements are not present, incomplete, or changing rapidly.

Elements of session-based testing

Mission

- The mission in Session Based Test Management identifies the purpose of the session, helping to focus the session while still allowing for exploration of the system under test.
- According to Jon Bach, one of the co-founders of the methodology, the mission tells us "what we are testing or what problems we are looking for.

Charter

- A charter is a goal or agenda for a test session. Charters are created by the test team prior to the start of testing, but they may be added or changed at any time.
- Often charters are created from a specification, test plan, or by examining results from previous sessions.

Session

- An uninterrupted period of time spent testing, ideally lasting one to two hours.
- Each session is focused on a charter, but testers can also explore new opportunities or issues during this time.
- The tester creates and executes tests based on ideas, heuristics or whatever frameworks to guide them and records their progress. This might be through the use of written notes, video capture tools or by whatever method as deemed appropriate by the tester

Session report

The session report records the test session. Usually this includes:

- Charter.
- Area tested.
- Detailed notes on how testing was conducted.
- A list of any bugs found.
- A list of issues (open questions, product or project concerns)
- Any files the tester used or created to support their testing
- Percentage of the session spent on the charter vs investigating new opportunities.
- Percentage of the session spent on:
  - Testing - creating and executing tests.
  - Bug investigation / reporting.
  - Session setup or other non-testing activities.
- Session Start time and duration.

3.7.8 Recovery Testing

GQ. Write a note on Recovery Testing.

- Recovery testing is done to verify that the data is recovered if any fault occurs; application crashes or power goes off.
- To test the recovery of data, the software is forcefully failed to check if data is recovered successfully. **E.g.** Printer can be disconnected to check if the system hangs or can be shut down to check data loss.

3.7.9 Installation Testing

- Installation testing is one of the most important testings as Installation is the very first interaction of the user with the product and it's important that the user does not face difficulty in installing the same.
- The software can be installed via CD, Internet, and network location.

Example : If an installation is done through the Internet, then test cases should be included for:

- Bad network speed
- Broken connection.
- Size and approximate time taken.
- Concurrent installation/downloads.
- The Installation of Software requires the use of License keys. Thus different types of licensing are:
  - Node-Locked
  - Floating
  - Named User
  - Temp/Evaluation Licenses

3.7.10 Ad hoc Testing

GQ. Explain Ad hoc Testing.

- **Ad hoc Testing** is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage.
- Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.
- Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application.
- Main aim of this testing is to find defects by random checking.

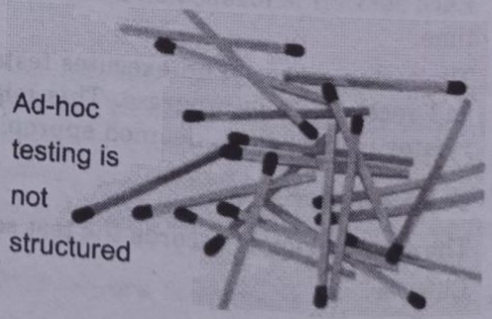


Fig. 3.7.3

- Adhoc testing can be achieved with the Software testing technique called **Error Guessing.** Error guessing can be done by the people having enough experience on the system to "guess" the most likely source of errors.
- This testing requires no documentation/ planning /process to be followed.
- Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

GQ. When execute Adhoc Testing ?

- Ad hoc testing can be performed when there is limited time to do elaborative testing.
- Usually adhoc testing is performed after the formal test execution. And if time permits, ad hoc testing can be done on the system.
- Ad hoc testing will be effective only if the tester is knowledgeable of the System Under Test.

☞ **Types of Adhoc testing**

There are different types of Adhoc testing and they are listed as below :

Buddy Testing	Two buddies mutually work on identifying defects in the same module. Mostly one buddy will be from development team and another person will be from testing team. Buddy testing helps the testers develop better test cases and development team can also make design changes early. This testing usually happens after Unit Testing completion.
Pair testing	Two testers are assigned modules, share ideas and work on the same machines to find defects. One person can execute the tests and another person can take notes on the findings. Roles of the persons can be a tester and scriber during testing. *Comparison Buddy and Pair Testing:* Buddy testing is combination of unit and System Testing together with developers and testers but Pair testing is done only with the testers with different knowledge levels. (Experienced and non-experienced to share their ideas and views)
Monkey Testing	Randomly test the product or application without test cases with a goal to break the system.

☞ **Best practices of Adhoc testing**

Following best practices can ensure effective Adhoc Testing

1. **Good business knowledge**
   - Testers should have good knowledge of the business and clear understanding of the requirements- Detailed knowledge of the end to end business process will help find defects easily.
   - Experienced testers find more defects as they are better at error guessing.
2. **Test Key Modules**
   - Key business modules should be identified and targeted for ad-hoc testing.
   - Business critical modules should be tested first to gain confidence on the quality of the system.
3. **Record Defects**
   - All defects need to be recorded or written in a notepad.
   - Defects must be assigned to developers for fixing. For each valid defect, corresponding test cases must be written and must be added to planned test cases.
   - These Defect findings should be made as lesson learned and these should be reflected in our next system while we are planning for test cases.

3.7.11 Risk Based Testing

GQ. Explain Risk Based Testing with example.

- **Risk based testing is basically a testing done for the project based on risks.** Risk based testing uses risk to prioritize and emphasize the appropriate tests during test execution.
- In simple terms – Risk is the probability of occurrence of an undesirable outcome. This outcome is also associated with an impact. Since there might not be sufficient time to test all functionality, Risk based testing involves testing the functionality which has the highest impact and probability of failure.

- Risk-based testing is the idea that we can organize our testing efforts in a way that reduces the residual level of product risk when the system is deployed.
  - Risk-based testing starts early in the project, identifying risks to system quality and using that knowledge of risk to guide testing planning, specification, preparation and execution.
  - Risk-based testing involves both mitigation – testing to provide opportunities to reduce the likelihood of defects, especially high-impact defects – and contingency – testing to identify work-around to make the defects that do get past us less painful.
  - Risk-based testing also involves measuring how well we are doing at finding and removing defects in critical areas.
  - Risk-based testing can also involve using risk analysis to identify proactive opportunities to remove or prevent defects through non-testing activities and to help us select which test activities to perform.
- The goal of risk-based testing cannot practically be – a risk-free project. What we can get from risk-based testing is to carry out the testing with best practices in risk management to achieve a project outcome that balances risks with quality, features, budget and schedule.

GQ. How to perform risk based testing ?

1. Make a prioritized list of risks.
2. Perform testing that explores each risk.
3. As risks evaporate and new ones emerge, adjust your test effort to stay focused on the current crop.

3.7.12 I18N Testing

GQ. Explain I18N Testing.

- Internationalization testing is a non-functional testing technique. It is a process of designing a software application that can be adapted to various languages and regions without any changes.
- Localization, internationalization and globalization are highly interrelated.

1. Localization (l10n)

- Localization is the process of adapting a product or content to a specific locale or market.
- Localization is sometimes written as **l10n**, where 10 is the number of letters between l and n.
- The aim of localization is to give a product the look and feel of having been created specifically for a target market, no matter their language, culture, or location.

2. Internationalization (i18n)

- Internationalization is a design process that ensures a product (usually a software application) can be adapted to various languages and regions without requiring engineering changes to the source code.
- Think of internationalization as readiness for localization.
- Internationalization is often written **i18n**, where 18 is the number of letters between i and n in the English word.

3. Globalization (g11n)

- Globalization, in the context of the language industry, refers to a broad range of processes necessary to prepare and launch products and activities internationally.
- Sometimes written **g11n**, globalization is an all-encompassing concept which applies to activities such as multilingual communication, global-readiness of products and services, and processes and policies related to international trade, commerce, education, and more.

3.7.13 L1ON Testing

GQ. Explain L1ON Testing.

- The world is shrinking with technology, and more and more enterprises are rolling out products across geographies to address global customer base. However, enterprises need the confidence that their product will meet the language and functionality requirements of the local users in the specific regions.
- At the same time, they also need the right approach to scale operations optimally across geographies and deliver product in the shortest possible time-to-market.
- The acceptance of an application rests primarily on how its users perceive it. Critical to that success is the applications' ability to integrate seamlessly with the native language and the cultural landscape of the target market.
- MWIDM brings you Globalization Testing Services to help rapid product rollout across geographies.
- We have built an exhaustive QA methodology to identify specifically potential issues and problems in the globalization process.
- We have specific local checklists for various countries to ensure exhaustive coverage during localization testing.
- We leverage our proven experience in successful globalization testing engagements, skilled multi-lingual QA talent pool, and globalization testing best practices to ensure quick testing for scalability of products.
- The globalization testing includes localization (L10N) and internationalization (I18N) testing. The various aspects of L10N and I18N testing are as shown in Fig. 3.7.4.

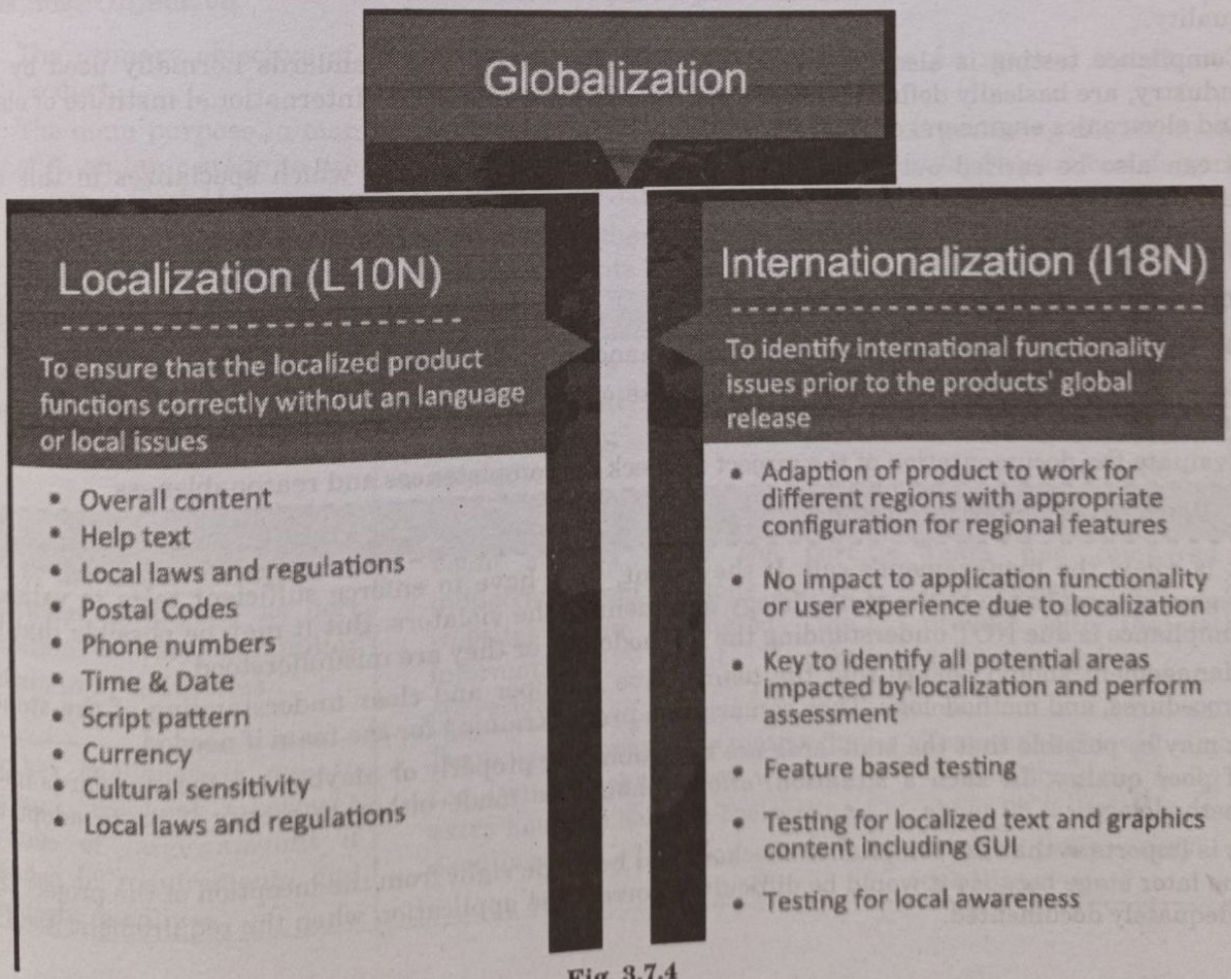


Fig. 3.7.4

☞ **Our Globalization Testing Differentiators**

- Rich experience in globalization testing supporting multi-country rollouts, giving you predictable results
- Localization-specific test optimization checklist and best practices to ensure maximum test coverage
- Specialized multi-lingual QA team with specific local knowledge for various countries
- Ability to reduce testing cycle time and enable quicker time-to-market for your products

3.7.14 Compliance Testing

GQ. Explain Compliance Testing.

☞ **Definition – What is Compliance Testing ?**

- "***Compliance testing***" also known as Conformance testing is a non-functional testing technique which is done to validate, whether the system developed meets the organization's prescribed standards or not.
- There is a separate category of testing known as "Non-Functional Testing".
- This is basically a kind of an audit which is done on the system to check if all the specified standards are met or not. To ensure that the compliances are met, sometimes a board of regulators and compliance expert people are established in every organization. This board puts a check whether the development teams are meeting the standards of the organization or not.
- The teams do an analysis to check that the standards are properly enforced and implemented. The regulatory board also works simultaneously to improve the standards, which will, in turn, lead to better quality.
- Compliance testing is also known as Conformance testing. The standards normally used by the IT industry, are basically defined by the large organizations like IEEE (International institute of electrical and electronics engineers) or W3C (World Wide Web Consortium), etc.
- It can also be carried out by an independent/third party company which specializes in this type of testing and service.

☞ **Objectives**

Objectives of compliance testing include

- Determining that the development and maintenance process meets the prescribed methodology.
- Ensures whether the deliverables of each phase of the development meets the standards, procedures, and guidelines.
- Evaluate the documentation of the project to check for completeness and reasonableness

GQ. When to use Compliance Testing ?

- It is solely the management's call. If they want, they have to enforce sufficient tests to validate the degree of compliance to the methodology and identify the violators. But it may be possible that lack of compliance is due NOT understanding the methodology or they are misunderstood.
- Management should ensure that the teams have a proper and clear understanding of the standards, procedures, and methodology. They can arrange proper training for the team if needed.
- It may be possible that the standards are not published properly or maybe that the standards itself are of poor quality. In such a situation, efforts should be made either to rectify it or to adopt a new methodology.
- It is important that the compliance check should be made right from the inception of the project than at the later stage because it would be difficult to correct the application when the requirement itself is not adequately documented.

GQ. How to do a compliance check ?

- Doing Compliance check is quite straight forward. A set of standards and procedures are developed and documented for each phase of the development lifecycle.
- Deliverables of each phase need to compare against the standards and find out the gaps. This can be done by the team through the inspection process, but I would recommend an independent team to do it.
- After the end of the inspection process, the author of each phase should be given a list of non-compliant areas that needs to correct.
- The inspection process should again be done after the action items are worked upon, to make sure that the non-conformance items are validated and closed.

#Exemplar/Case Studies

1. **Case Study : Manual Testing (Online Marketing Software Platform)**

Background

- The Client – INTECH Consultancy, is one of growing IT Company in India that provides software solution for Logistics and Marine for Hong Kong based clients, it also provides other services to small and medium businesses. They also providing the customer support for their shipment based products.
- The main product for client is based on Port Operation Management which contains shipment and vessel operations.

Business Objective

- The primary objective of KiwiQA team was to provide the solution for manual testing in their product.
- The main purpose in manual testing is to understand the product and gathering the requirements of it, on later stage to prepare the business scenarios or the test cases for the product, logging the issues in Test Management tool, prepare test data in spreadsheet for the test execution.
- Apart from this objective, team provide the other solutions like regression testing, smoke testing for each releases and suggesting the improvements in product.

Challenges	KiwiQA Approach	Value Delivered
• Understanding the port operation management flow.	• Team gathers the requirement from client and take understanding of the port operations.	• Team prepared around 3800 + test cases (75 % functional and 25 % Usability/UI) and logged around 1500 + bugs in the product.
• One of the major challenge is functional complexity as port operation is very unfamiliar domain in IT industries.	• Team studied about the port industries from various online websites and gathered the information about port operations.	• Team has compared and understand the client product with the gathered documents online.
• Another challenge is due to having frequent releases because of large amount if changes in requirements and very tight deadlines.	• Team used traceability matrix for the issue tracking and working extra hours to achieve the testing deadlines.	• Team has achieved almost 90 % to deliver the build with about 95 % accuracy in every release with very tight deadlines.

Solution Background and Engagement Details- The Outcome

- Our manual testing solution was delighted and given the confidence to the Customer where it had functional improvements, efforts and accuracy.
- In earlier stages it was a bit tricky to work on the project and challenging as well to understand the complete flow of port operation management.
- On later stages once the team gathered the requirements and understood the product, the test execution approach gets better and the execution time decreased.
- Our solution has gained client confidence and thus strengthens the relations between us and which in turn made client to give more work over a longer period of time.
- Link:https://www.360logica.com/blog/case-study-manual-testing-online-marketing-software-platform/

2. Case Study : Decision Table Testing (transferring money online to an account which is already added and approved.)

- Decision Table Testing is a software testing methodology used to test system behavior for various input combinations.
- In this systematic approach, the several input combinations and their corresponding system behavior are represented in tabular form.
- The decision table is also called a Cause-Effect table, as the causes and effects for comprehensive test coverage are captured in this table.
- Decision Table testing is a commonly used black-box testing technique and is ideal for testing two or more inputs that have a logical relationship.

GQ. What is a Decision Table ?

- A decision table is the tabular representation of several input values, cases, rules, and test conditions.
- The Decision table is a highly effective tool utilized for both requirements management and complex software testing. Through this table, we can check and verify all possible combinations of testing conditions.
- The testers can quickly identify any skipped needs by reviewing the True(T) and False(F) values assigned for these conditions.

☞ **Advantages Of Decision Table Testing**

1. Decision tables are one of the most effective and full-proof design testing techniques.
2. Testers can use decision table testing to test the results of several input combinations and software states.
3. It gives the developers to state and analyzes complex business rules.
4. Decision table testing is the most preferred black box testing and requirements management.
5. A decision table is used for modeling complex business logic. They can first be converted to test cases and test scenarios through decision table testing.
6. This technique provides comprehensive coverage of all test cases that can significantly reduce the re-work on writing test cases and test scenarios.
7. Decision tables guarantee coverage of all possible combinations of condition values which are called completeness property.
8. Decision tables can be used iteratively. The table results created in the first testing iteration can be used for the next and so on.

9. Decision tables are easy to understand, and everyone can use and implement this design and testing method, scenarios and test cases without prior experience.
10. Multiple conditions, scenarios and results can be viewed and analyzed on the same page by both developers and testers.

GQ. Give short note on : Decision Table Examples.

A decision table is a tabular representation of inputs vs cases, rules and test conditions.

Example 1 : In this example, we see how to create the decision table for a login screen that asks for UserId and Password.

The condition here is that the user will be redirected to the homepage if he enters the correct user name and password, and an error message will be displayed if the input is wrong.

Conditions	Rule 1	Rule 2	Rule 3
Username (T/F)	F	T	F
Password (T/F)	F	F	T
Output (E/H)	E	E	E

Legend

T – Correct username or password
F – Wrong username or password
E – Error message is displayed.
H – Home screen is displayed.

☞ **Decision Table Interpretation**

- **Case 1 :** Username and Password both are wrong, and the user is shown an error message.
- **Case 2 :** Username is correct, but the password is wrong, and the user is shown an error message,
- **Case 3 :** The username is wrong, but the password is correct, and the user is shown an error message.
- **Case 4 :** Username and password both are correct, and the user is taken to the homepage.

☞ **Test Scenarios Possible For This Decision Table**

1. Enter correct username, correct password, and click on login. The expected result is that the user should navigate to the homepage.
2. Enter correct username, wrong password, and click on login. The expected result is that the user should get an error message.
3. Enter the wrong username, correct password, and click on login. The expected result is that the user should get an error message.
4. Enter the wrong username, wrong password, and click on login. The expected result is that the user should get an error message.

Example 2 : In this example, we consider the decision table and test scenarios for an Upload screen.

There is a dialogue box that will ask the user to upload a photo with the following conditions:

The file must be in the .jpg format.

The file size must be less than 32kb.

The image resolution must be 137*177.

If any one of the above conditions fails, the system will display the corresponding error messages about the issue. If all conditions are satisfied, the photo will be uploaded successfully.

Conditions	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Format	.JPG	.JPG	.JPG	.JPG	Not.JPG	Not.JPG	Not.JPG	Not.JPG
Size	< 32 kb	< 32 kb	>= 32 kb	>=32 kb	< 32 kb	< 32 kb	>= 32 kb	>= 32 kb
Resolution	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177
Output	Photo uploaded successfully	Error message due to resolution mismatch	Error message due to size mismatch	Error message due to size and resolution mismatch	Error message due to format mismatch	Error message due to format and resolution mismatch	Error message due to format and size mismatch	Error message due to format, size and resolution mismatch

For these conditions of the decision table, we can formulate eight different test cases or input scenarios to cover all the possibilities.

☞ Scope of Decision Table Testing

- When data is complex, and every combination needs to be tested, decision tables can become huge.
- You can intelligently reduce the number of varieties in each possibility to only choose the interesting and impactful ones. This approach is called Collapsed Decision Table Testing.
- In this technique, redundant conditions that are irrelevant to the outcome are removed, and different outputs are produced. An additional layer of analysis is added to the test design so that the tester can perform more effective testing.
- Decision tables are a robust specification-based testing technique that can work for many scenarios.
- The tabular and the graphical representation is very beneficial for all stakeholders and non-technical members to understand easily.
- The project team members can instantly obtain detailed insights about the problem at hand through illustrative examples and real-life scenarios.
- By moving to the next level of collapsed decision-making table, the management can realize the effectiveness and efficiency of this testing technique.

...Chapter Ends

❑❑❑

UNIT IV

CHAPTER 4

Software Quality Assurance and Quality Control

University Prescribed Syllabus for the Academic Year 2022-2023

Software Quality Assurance : Introduction, Constraints of Software Product Quality Assessment, Quality and Productivity Relationship, Requirements of a Product, Characteristics of Software, Software Development Process, Types of Products, Schemes of Criticality Definitions, Software Quality Management, Why Software Has Defects? Processes Related to Software Quality, Quality Management System Structure, Pillars of Quality Management System, Important Aspects of Quality Management.

Software Quality Control : Software quality models, Quality measurement and metrics, Quality plan, implementation and documentation, Quality tools including CASE tools, Quality control and reliability of quality process, Quality management system models, Complexity metrics and

Customer Satisfaction, International quality standards – ISO, CMM

Rohit Patil

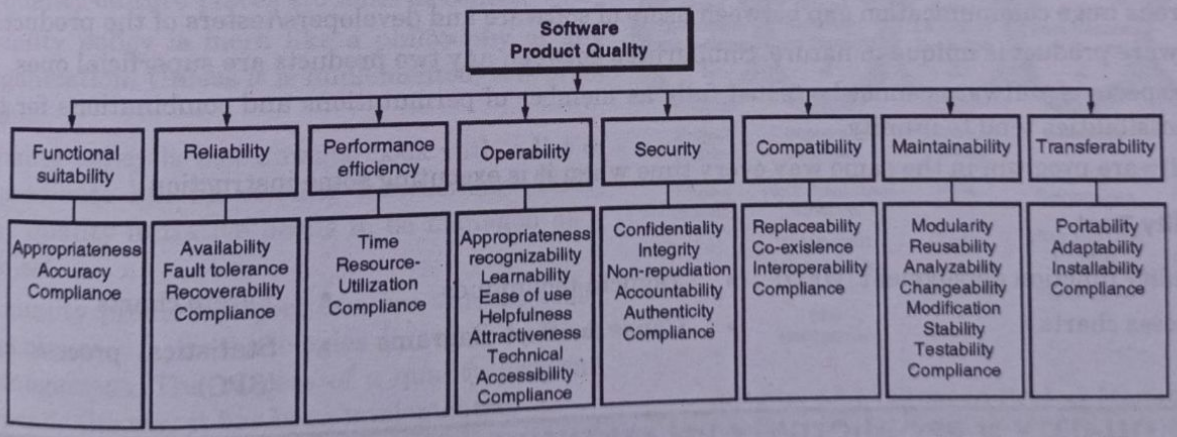
4.1 SOFTWARE QUALITY ASSURANCE -INTRODUCTION

The degree to which a scheme, component, or process chances definite requirements.

1. Quality is suitability for use.
2. Conformance to specification.
3. Transparency of service delivery
4. Achieving desired results
5. Continuous Improvement
6. Competitive advantage
7. Added value for society Best value for price
8. Cost effectiveness
9. Performance measurement
10. Satisfaction of stakeholders
11. Doing the correct things
12. Doing things right & doing the right effects right

4.1.1 Quality in Software Engineering

- In broader terms, the software quality definition of "fitness for purpose" refers to the satisfaction of requirements. But what are requirements? Necessities, also called user levels in today's agile terms, can be considered as practical and non-functional. Functional requirements mention to precise functions that the software should be able to complete.
- For example, the facility to print on an HP Inkjet 2330 printer is a useful prerequisite. However, just since the software has a confident function or a user can comprehensive a task consuming the software, does not mean that the software is of respectable quality.
- There are probably many instances where you've used software and it did what it was supposed to do, such as find you a flight or make a hotel reservation, but you thought it was poor quality. This is because of how the function was executed. The displeasure with "how" signifies the non-functional necessities not being met.
- For this purpose the International Organization for Standardization (ISO) developed as a model for specifying non-functional necessities. The classical shown below exemplifies the classification of non-functional requirements.



(1A10) **Fig. 4.1.1 : Software product Quality Requirements and Evaluation (SQuaRE) Quality model and guide**

Why Non-Functional Quality Components Are Important ?

Adequate non-functional necessities such as presentation, ease of use and learn aptitude first necessitates requiring and important. Only then can they be contented, and sufficient them can be even more problematic than substantial functional necessities. So, what does this mean for you? Let's examine the following non-functional characteristics using the same printing example :

- **Functional Suitability (functional appropriateness)** Does the function facilitate the completion of the user's task(s) and objectives? If the user doesn't want to print on that printer or wants to print a PDF but isn't given those options, then maybe not.
- **Performance Efficiency (time behaviour)** – Does the printer purpose return inside three seconds?
- **Compatibility(interoperability)** – Can the operator print over a variety of networks and printers and on processors with unlike operative schemes(Windows and Mac)?
- **Usability (learnability)** – Can the operative figure out in what way to print or will it income a rocket expert?
- **Consistency (recoverability)** – When the printer is released in the internal of printing a task, is the user learned?
- **Security (nonrepudiation)** – Is contiguous a record that the printer published the file efficiently?
- **Maintainability (testability)** – Can test events be measured for the print function?
- **Portability (adaptability)** – Can the software mechanically familiarize to new printer models, or an update in printer driver software ? Can the print purpose deliver shortcuts for highly refined users?

Now that we have an accepting of non-functional requests, let's examine the excellence lifecycle in the diagram beneath. Looking at the three circles under, internal excellence represents quality that you wouldn't realize and is stately by internal possessions such as code feature. Exterior worth signifies what we have conversed above in the non-functional superiority model and is typically measured by actual execution of the code and examination of software behaviour.

4.2 CONSTRAINTS OF SOFTWARE PRODUCT QUALITY ASSESSMENT

Requirement specification is made by business analyst and system analyst. Tester may or may not have direct access to the customer and may get information though requirement statements, queries answered etc. either from customer or business analyst. There are few limitations of product quality assessment in this.

- Software is virtual in nature. Software products cannot be touched or heard.
- There is huge communication gap between users of software and developers/testers of the product.
- Software product is unique in nature. Similarities between any two products are superficial ones.
- All aspects of software cannot be tested fully as member of permutations and combinations for testing all possibilities tend to infinity.
- A software program in the same way every time when it is executing some instruction.

Quality Tool

- Quality function deployment (QFD)
- Process charts
- Taguchi techniques
- Cause & effect diagrams
- Pareto charts
- Statistical process control (SPC)

4.3 QUALITY & PRODUCTIVITY RELATIONSHIP

- Productivity is the relationship between a given amount of output and the amount of input needed to produce it. Quality affects productivity.
- Productivity is tool of quantity that defines the competence of the association in relations of the ratio of output created with esteem to inputs used.
- Quality must improve productivity by reducing wastage.
- Improvement in quality directly leads to improved productivity.

- Cost reduction is possible by improved quality.
- Proper communication between management and employee is essential.
- Quality improvement leads to cost reduction.
- Employee involvement in quality improvement.

4.4 REQUIREMENTS OF PRODUCT

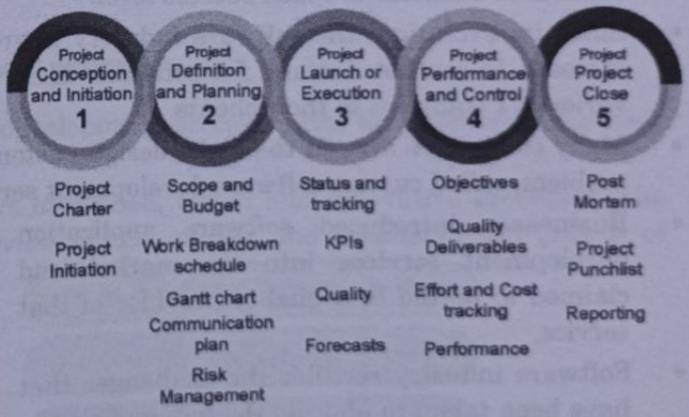
1. **Stated/Implied Requirements:** Functional and non-functional requirements stated by customer.
2. **General/Specific Requirement :** Requirements are generic in nature.
3. **Present/Future Requirements :** Present when application is used and future for required after some time span.
4. **Primary Requirements:** Must /must not be requirements. Customer pay for this
5. **Secondary Requirements:** Should/Should not be requirements.
6. **Tertiary Requirements:** Could/could not be requirements.

4.5 CHARACTERISTICS OF SOFTWARE

Quality culture is set of group standards that guide how developments are made to average working practices and resultant outputs. An organization's values can support entities at all levels make improved and more accountable choices involving issues of quality.

4.5.1 Following Features Emerged as Indicative of a Quality Culture Project Management

- Academic Ownership of quality.
- Quality culture is primarily about the behavior of stakeholders rather than the operation of a quality.
- A quality culture places students as center.
- Quality policy is more like a philosophy of any organization. Unless it is implemented, it is of no value.
- Quality policy is like a recipe book with a list of ingredients but no cooking instructions. Thus, any quality initiative needs to be managed as a project.
- A quality product project can use various tools, techniques, methodologies of project management. The success of a quality program depends the way it has been implemented.



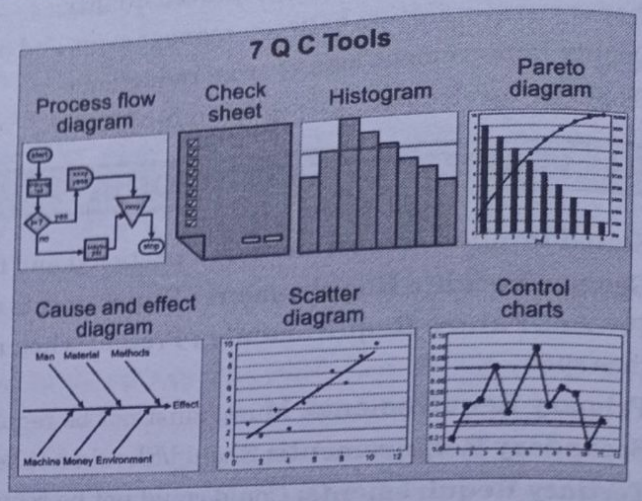
(1A11)**Fig. 4.5.1 : Steps in Project Management**

4.5.2 Sustainability and Quality Management

- Process efficiency, quality metrics, reduced waste etc. have started fascinating Sustainability practitioners nowadays. Sustainability teams are undergoing training & certifications to make themselves conversant with quality management.
- They are keen to use various tools & techniques of quality management in sustainability.

Seven Quality Control Tools

- Like Quality, Sustainability also has a strong attention on people. It takes into account quality of working life and employee satisfaction also.
- ISO 26000 brands a more thoughtful joining between people and excellence organization systems.
- Quality management is going to create value in Sustainability space in a big way.
- Thus, the trend of convergence between quality management and Sustainability in the offing.



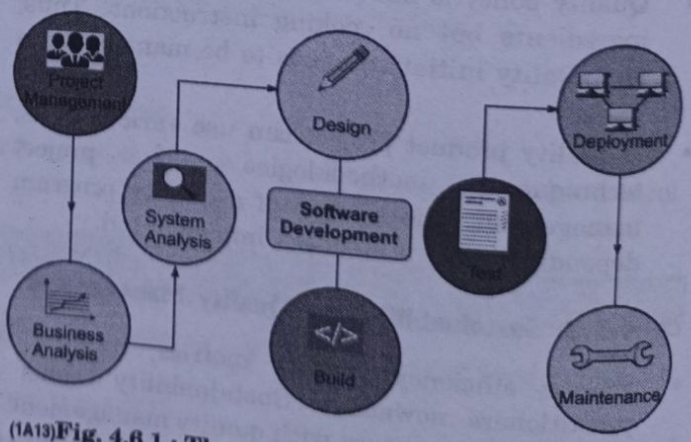
(1A12)**Fig. 4.5.2 : 7 QC Tools**

4.6 SOFTWARE DEVELPOMENT PROCESS

UQ. What is impact of defect in different phases of software development ?

SPPU - Nov./Dec.19 (End Sem)

- When we talk about the software development process, it requires establishing a connection between the idea and the approach to implement that idea.
- Thus, a proper process needs to be followed while implementing that idea for software development process. Over the years, the evolution of software development process came into existence for building customers business to reach success level.
- Since its evolution, the software industry is growing at a very fast pace because of the global level increase in market demand for software. Many players entered in the software industry and offered various IT solutions to their clients.
- Many companies around the globe design custom software to address their client's particular business problem, called custom software development services.
- Businesses introduced software application development services into the market and claimed to be the best quality provider of that service.
- Software industry rectifies the challenges that have been taken to identify the gap and risk in managing software development.
- These steps are taken to confirm the quality of software processing at each level. It basically refers to attempt for creating the software application to meet the need of client's business.



(1A13)**Fig. 4.6.1 : The process of Software Development**

4.6.1 Different Phases Included in Software Development Process

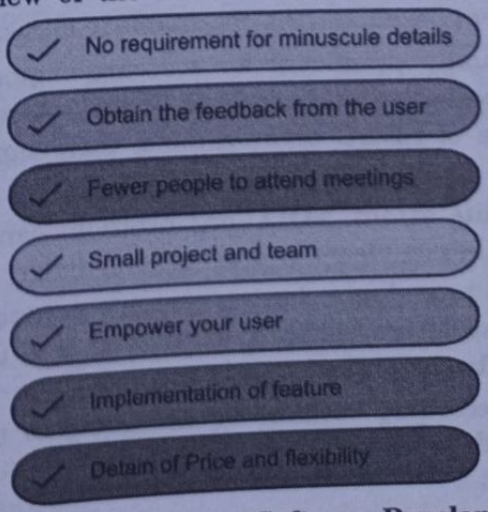
When it emanates to software development services, there are several approaches and models included in it. Here, we are conversing few major periods of software development methodologies :

Phases in Software Development

1. **Considerate the requirement :** In order to grow fully useful software, one necessity obviously appreciate the necessities of its customers. This is the most significant concern that should be taken, before start planning & working on the entire development process. A developer faces many challenges & alterations before coming up with the final plan as per the requirement & brief shared by the customer.
2. **Feasibility Analysis :** This phase involves scrutinizing the project whether it is feasible to work or not. Hence, you must build a strong interconnection between the project requirements and the need of your customer's project.
3. **Design :** Design plays a very imperative role in attracting visitors and generating more traffic. This includes production the complete architecture of the software. Henceforth, the design must be formed highly creatively and exclusively so that regulars can get best consequences.
4. **Coding :** This period contracts with the designer or programmers. It completely be contingent upon customer's choice in which programming language do they need to establish their project. It involves the transformation of design into coding through the help of a programmer.
5. **Software Testing :** Once the code is generated; it undergoes through various testing phases. This determines whether the product established is original or not. At this phase, any kind of bug or glitches found can be fixed.
6. **Maintenance :** Last but not the least, high maintenance is required in the project before & after it is being delivered to the user. The developer checks if the software is working fine before delivering it to the users & provides after sales service support & assistance in relations of the working of the software to the end users.

4.6.2 Seven Ways to Develop Software Development Process

- There are some ways through which the software development processes can be improved effectively such as:-
- Evolution in software development came with years of refined, tested and innovative processes. The demand for software development process has a view of the market and results in the growth of businesses.
- Life has never been better, therefore software provides human with the technology that helps in better and efficient accomplishment of tasks, functions, and goals.
  - No obligation for minuscule detail
  - Obtain the feedback from the user
  - Fewer people to attend meetings
  - Small projects and team
  - Empower your user
  - Implementation of features
  - Detain of price & flexibility



(1A14) **Fig. 4.6.2 : Seven ways to develop Software Development Process**

- Following Software Development Process Models are used :
  1. Waterfall Development Approach/Model.
  2. Iterative Development Approach/Model
  3. Incremental Development Approach/Model
  4. Spiral Development Approach/Model
  5. Prototyping Development Approach/Model
  6. Rapid Development Approach/Model
  7. Agile Development Approach/Model

4.6.3 Types of Product

- Life Affecting Products.
- Product Affecting Huge Some of Money.
- Products Which can be Tested only by Simulators: Example Space Research
- Other Products.

4.7 SCHEMES OF CRITICALITY DEFINATIONS

From User's Perspective

Failure of product disrupts entire business. Products failure affects business.

From Developer's Perspective

This classification defines the complexity of the system on the basis of development capabilities required.

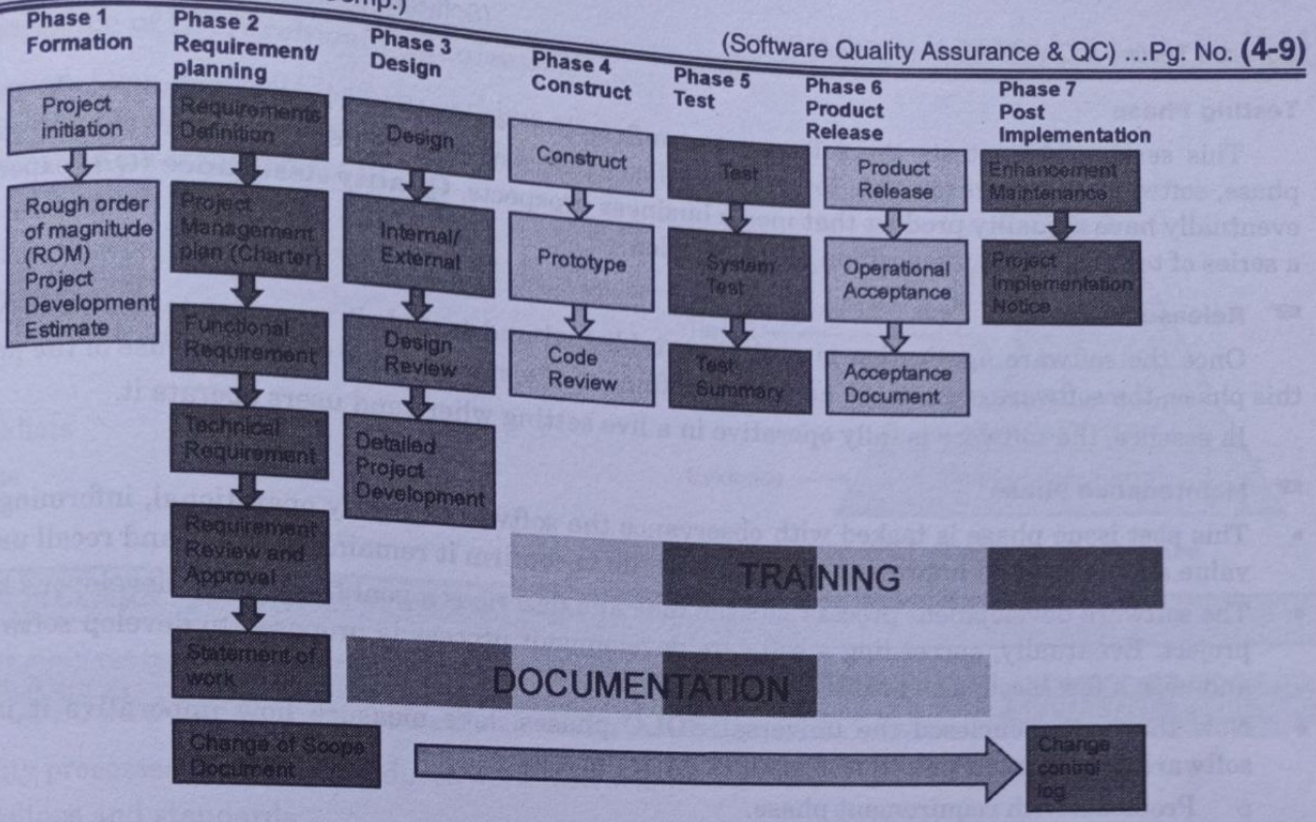
4.7.1 Problematic Areas of SDLC

Software Improvement Life Cycle Organizations

What is SDLC ?

- Every day, software engineers and specialists similar have to submerge themselves into the subtleties of the best **Software Development Lifecycle (SDLC)** procedure and method to progress and transport software in optimal environments. But, what is SDLC?
- In the meekest positions, SDLC practices deliver a methodical context to initiative, progress and offer software requests, from start to end. It is a sequence of phases that proposal a basis for the software growth process.
- Having a organization to grow software is important, which is why there are numerous software development organizations obtainable to choose from.
- It is gradually significant for software engineers to excellent the accurate **SDLC** model that meets precise necessities and concerns of the scheme to drive achievement. In this article, we go into the particulars of **SDLC** organizations, their significance, their benefits, difficulties, and everything in between.
- To a definite extent, **SDLC** organizations can be assumed of like a checklist of the dissimilar stages that must be completed to progress and deliver effective software requests.
- All **SDLC** procedures share a common ground of separate stages that include development, analysis, strategy, structure, difficult, organizing, and conservation. These **SDLC** stages provide the outline of what a software request project requires.

In the succeeding section, we are going to travel how software development lifecycles impact the software growth process.



(1A15)**Fig. 4.7.1 : SDLC phases**

☞ The Software Development Process

- The software development method, as with all excessive projects, starts with an idea. It earnings planning, preparation, and management of stages and team members to influence a goal.
- SDLC is a mapped out, planned context that classically surveys the following universal stages to transport high quality software application.

☞ Formation phase

This basic, initial phase is the beginning of an idea for a resolution that advances an existing solution or develops an entirely new one. It helps define the magnitude of the project to plan resources.

☞ Requirement/Planning Phase

In this phase, supplies are gathered to formulate a design plan for the software application solution. This phase involves a systematic examination to evaluate user requests, feasibility, growth, enhancements, and more. It is very significant to contain documentation to refine necessities and keep a record of the solution's development. This phase includes the formation of a project charter which describes technical and functional necessities.

☞ Design Phase

This phase is intensive on the design feature of the software application resolution in relations of the designated technical and functional necessities and the effects of the exhaustive enquiry of the software's feasibility.

☞ Development Phase

This phase is the meat of the software growth process. In this phase, software engineers are exclusively attentive on building an example of the solution to attain a code evaluation and finally create the solution itself. The team everything on converting software conditions into a employed and dependable solution.

Testing Phase

This serious phase tests the software to confirm that entirety works as it planned. In the analysis phase, software engineers are able to identify deficiencies, bacteria, and errors in the software solution and eventually have a quality product that meets business prospects. **Quality Assurance (QA)** experts perform a series of tests to assess the position of the solution.

☞ **Release Phase**

Once the software application is entirely established and tested, it transfers to the release phase. In this phase, the software goes aware and is unconfined to the end operator for definite use of the product.

In essence, the software is fully operative in a live setting where end users operate it.

☞ **Maintenance Phase**

- This post issue phase is tasked with observance the software entirely operational, informing it to meet value standards, and improving it through its life to confirm it remains to entice and recall users.
- The software development process sets the tone and describes a goal from which developer's kick-start a project. Eventually, succeeding a software development process is proposed to develop software earlier and with a few hiccups as possible.
- Now that we've enclosed the universal **SDLC** phases, let's measure how imperative it is to follow software development procedures in an IT environment.
  - Problems with requirement phase.
  - Requirements change very frequently.
  - Unique product is built in any time.
  - Product nature is intangible.
  - Inspection can be exhaustive.
  - Requirements are not easily communicated.
    - Technical Requirements.
    - Economical Requirements.
    - Operational Requirements.
    - System Requirements.

4.8 SOFTWARE QUALLITY MANAGEMENT

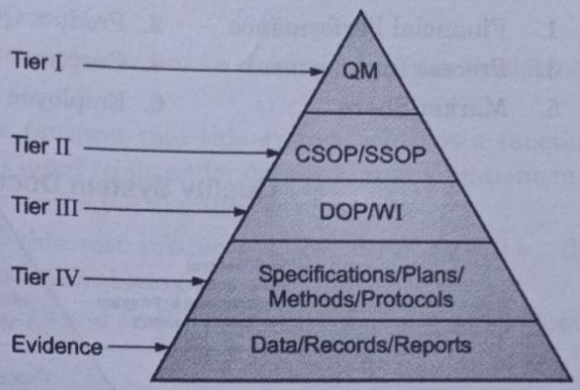
UQ. With respect Quality Management Systems explain the following ?
(i) Quality Management Systems Structure (ii) Pillars of Quality Management System
(iii) Important aspects of quality management **SPPU - Aug. 18(In Sem)**

- Quality management involves management of all inputs and processing defined so that the output from the process as per defined criteria.
- It handles three levels of problems :
  - Correction.
  - Corrective Actions.
  - Preventive Actions.
- Quality Management System Structure
  - 1st Tier-Quality policy
  - 2nd Tier-Quality objectives
  - 3rd Tier-Quality Manual

The Importance of Hierarchical Organization

An organization is spirited when working with controlled documents. A suggested hierarchy for QMS documentation management is:

1. Quality Manual
2. Policies
3. Procedures
4. Work Instructions
5. Lists
6. Checklists
7. Forms



(1A16)**Fig. 4.8.1 : QMS Tier structure**

4.9 PILLARS OF QUALITY MANAGEMENT SYSTEM

UQ. What are the pillars of Quality Management System ? **SPPU - Oct. 19 (In Sem)**

- Quality processes/Quality Procedures/work instructions
- Guidelines and standards
- Formats and Templates

☞ **Steps for the Creation of an Effective QMS**

The steps required for the conceptualization and implementation of a QMS include the following :

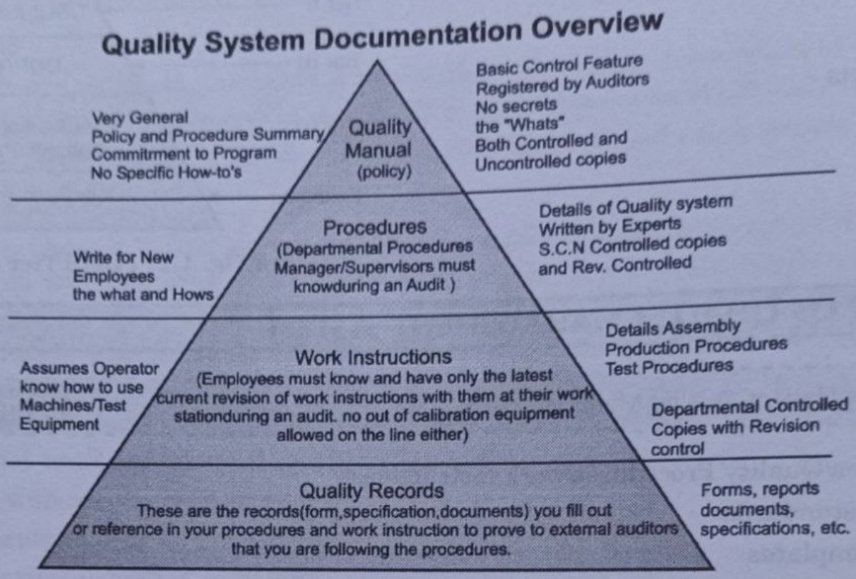
1. **Define and Map Your Processes**
   - Process maps creation will force the organization to visualize and define their processes. In the process, they will define the interaction sequence of those processes.
   - Process maps are vital for appreciating the responsible person. Define your main business process and converse the flow.
2. **Define Your Quality Policy**
   - Your Quality Policy communicates the duty of the organization as it is about the quality. The mission may be what customers need, a quality mission.
   - When constructing quality management system, consider the commitment towards customer focus. It may be Quality, Customer Satisfaction, and Continuous Improvement.
3. **Define Your Quality Objectives**

UQ. What are types of Requirements and Product? Also point out relationship between Quality and Productivity ? **SPPU - Aug 18(In Sem)**

All Quality management systems must have objectives. Each employee must appreciate their influence on quality. Quality objectives are derivative of your quality policy. It is measurable and set up throughout the organization. The objective may be in the form of critical success factors. This helps an organization in emphasizing the journey towards accomplishing its mission. These performance-based measures deliver a gauge to determine compliance with its objectives.

Some Critical Success Factors are:

1. Financial Performance
2. Product Quality
3. Process Improvement
4. Customer Satisfaction
5. Market Share
6. Employee Satisfaction



(1A17)**Fig. 4.9.1 : Quality System Documents (QSD) Overview Levels**

4. Develop Metrics to Track and Monitor CSF Data

- Once critical success factors are known, measurements and metrics keep track of advancement.
- This can be complete complete a data reporting procedure used to collect specific data. Share the processed information with leaders. A process goal is to improve customer approval index score.
- There requests to be a goal and a quantity to inaugurate realization of that goal.

4.10 IMPORTANT ASPECTS OF QUALITY MANAGEMENT

1. Quality planning at organisation level
2. Quality planning at project level
3. Resource management
4. Work Environment
5. Customer Related Processes
6. Quality management system document and data control
7. Verification and validation
8. Software project management
9. Software configuration management
10. Software metrics and measurement
11. Software Quality Audits

Software Quality Attributes are : Correctness, Reliability, Adequacy, Learn facility, Robustness, Maintainability, Readability, Extensibility, Testability, Competence, and Portability.

1. **Correctness :** The perfection of a software system mentions to:
   - Promise of program code with conditions
   - Individuality of the definite request of the software system.
   - The correctness of a program develops especially critical when it is implanted in a composite software system.

2. **Reliability :** Reliability of a software system develops from
(a) Correctness (b) Accessibility
- The conduct over time for the contentment of a given requirement be contingent on the consistency of the software system.
- Reliability of a software system is definite as the prospect that this system achieves a function (resolute by the provisions) for a definite number of input trials under detailed input situations in a definite time interval (presumptuous that hardware and input are allowed of errors).
- A software system can be seen as consistent if this test produces a low error rate (i.e., the possibility that an error will occur in a definite time interval.)
- The error rate is contingent on the frequency of inputs and on the possibility that a separate input will lead to an error.

3. **Acceptability :** Factors for the prerequisite of Capability:
- The input compulsory of the user should be imperfect to only what is essential. The software system should assume information only if it is essential for the purposes that the user needs to transmit out.
- The software system should allow plastic data input on the part of the user and should carry out credibility checks on the input. In dialog ambitious software systems, we vest specific standing in the consistency, clearness and ease of the interchanges.
- The presentation accessible by the software system should be altered to the needs of the operator with the thought given to extensibility; i.e., the purposes should be incomplete to these in the requirement.
- The results created by the software system : The outcomes that a software system distributes should be output in a clear and well organized form and be informal to understand.
- The software system should afford the user flexibility with respect to the scope, the degree of detail, and the form of presentation of the results. Error messages must be provided in a form that is comprehensible for the user.

4. **Learn ability :** Learn ability of a software system depends on:
- The design of user interfaces.
- The clarity and the simplicity of the user instructions (tutorial or user manual).
- The user interface should present information as close to reality as possible and permit efficient utilization of the software's failures.
- The user manual should be structured clearly and simply and be free of all dead weight.
- It should clarify to the user anything the software system should do, in what way the separate functions are motivated, what relations exist between functions, and which exclusions strength arise and how they can be modified.
- In adding, the user guide should serve as a orientation that maintenances the user in quickly and securely definition the correct responses to queries.

5. **Robustness :** Robustness decreases the impact of working mistakes, specious input data, and hardware errors.
- A software system is robust if the significances of an error in its process, in the input, or in the hardware, in relative to a given application, is contrariwise comparative to the possibility of the amount of this error in the given request.
- Frequent errors (e.g. erroneous commands, typing errors) must be touched with particular care.
- Less recurrent errors (e.g. power letdown) can be touched more laxly, but still must not lead to permanent consequences.

6. **Maintainability :** Maintainability = suitability for correcting (localization and correction of errors) and for modification and extension of functionality.
   The maintainability of a software system be contingent on its:
   - Readability
   - Extensibility
   - Testability

7 **Readability :** Readability of a software system be contingent on its:
   - Form of illustration
   - Programming style
   - Consistency
   - Readability of the application programming languages
   - Structureless of the system
   - Quality of the certification
   - Tools available for inspection

8. **Extensibility :** Extensibility sanctions compulsory alterations at the suitable locations to be made without undesirable side effects. Extensibility of a software system depends on its :
   - Structureless (modularity) of the software system
   - Opportunities that the application language delivers for this resolution
   - Readability (to find the suitable location) of the code
   - Availability of comprehensible program documentation

9. **Testability :** appropriateness for permitting the programmer to survey program implementation (runtime performance under given conditions) and for correcting.

10. **The testability** of a software system be contingent on its Modularity Structureness Modular; well-structured databases prove more appropriate for systematic, stepwise difficult than monumental, structured programs.

11. **Testing tools** and the opportunity of expressing constancy situations (declarations) in the source code decrease the testing effort and afford important fundamentals for the general, organized testing of all system mechanisms.

12 **Efficiency :** ability of a software system to accomplish its determination with the best possible operation of all essential properties (time, storage, transmission channels, and peripherals).

13. **Portability :** the easiness with which a software system can be altered to run on computers other than the one for which it was intended.

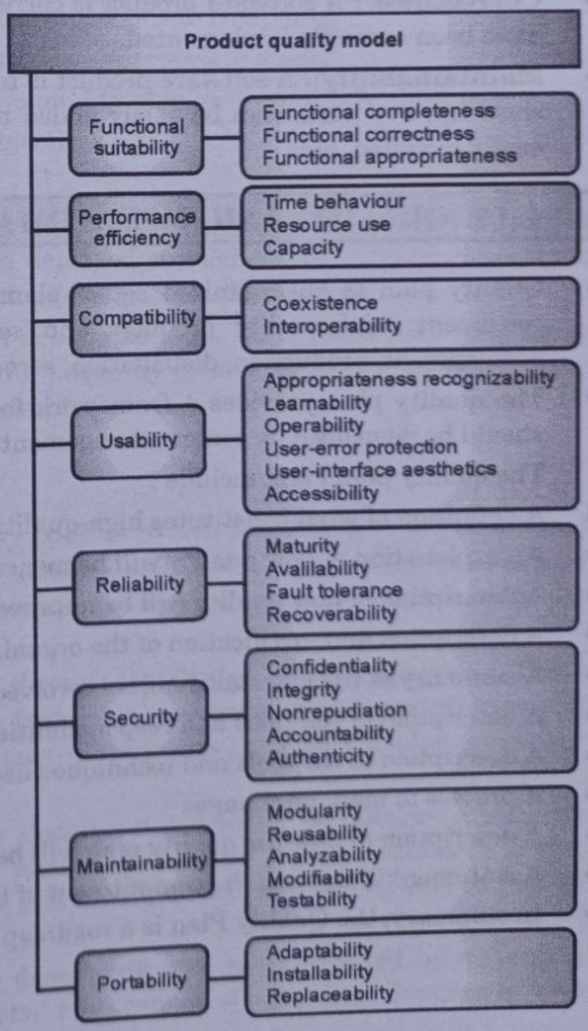
14. The movability of a software system be contingent on:
   - Degree of hardware independence
   - Implementation language
   - Magnitude of manipulation of dedicated system functions
   - Hardware possessions
   - Structureless : System needy elements are composed in easily substitutable program mechanisms.
   - A software system can be said to be transferable if the effort required for porting it shows meaningfully less than the effort essential for a new implementation.

4.11 SOFTWARE QUALITY CONTROL

- Quality control popularly abbreviated as QC. It is a Software Engineering process used to ensure quality in a product or a service. It does not deal with the processes used to create a product; rather it examines the quality of the "end products" and the final outcome.
- The main aim of Quality control is to check whether the products meet the specifications and requirements of the customer. If an issue or problem is identified, it needs to be fixed before delivery to the customer.
- QC also evaluates people on their quality level skill sets and imparts training and certifications. This evaluation is required for the service-based organization and helps provide "perfect" service to the customers.

4.12 SOFTWARE QUALITY MODEL

- This approach to software quality is best exemplified by fixed quality models, such as ISO/IEC 25010:2011. This standard describes a hierarchy of eight quality characteristics, each composed of sub-characteristics :
  1. Functional suitability
  2. Reliability
  3. Operability
  4. Performance efficiency
  5. Security
  6. Compatibility
  7. Maintainability
  8. Transferability
- Additionally, the standard defines a quality-in-use model composed of five characteristics :
  1. Effectiveness
  2. Efficiency
  3. Satisfaction
  4. Safety
  5. Usability
- A fixed software quality model is often helpful for considering an overall understanding of software quality.
- In practice, the relative importance of particular software characteristics typically depends on software domain, product type, and intended usage. Thus, software characteristics should be defined for, and used to guide the development of, each product.
- Quality function deployment provides a process for developing products based on characteristics derived from user needs.
- Software quality product is defined in term of its fitness of purpose. That is, a quality product does precisely what the users want it to do. For software products, the fitness of use is generally explained in terms of satisfaction of the requirements laid down in the SRS document.



(1E2) **Fig. 4.12.1 : Software Quality Model**

- Although "fitness of purpose" is a satisfactory interpretation of quality for many devices such as a car, a table fan, a grinding machine, etc. For software products, "fitness of purpose" is not a wholly satisfactory definition of quality.
- **Example :** Consider a functionally correct software product. That is, it performs all tasks as specified in the SRS document. But, has an almost unusable user interface. Even though it may be functionally right, we cannot consider it to be a quality product.
- The modern view of a quality associated with a software product several quality methods such as the following:

4.13 QUALITY MEASUREMENT AND METRICS

- **Portability :** A software device is said to be portable, if it can be freely made to work in various operating system environments, in multiple machines, with other software products, etc.
- **Usability :** A software product has better usability if various categories of users can easily invoke the functions of the product.
- **Reusability :** A software product has excellent reusability if different modules of the product can quickly be reused to develop new products.
- **Correctness :** A software product is correct if various requirements as specified in the SRS document have been correctly implemented.
- **Maintainability :** A software product is maintainable if bugs can be easily corrected as and when they show up, new tasks can be easily added to the product, and the functionalities of the product can be easily modified, etc.

4.14 QUALITY PLAN IMPLEMENTATION AND DOCUMENTATION

- Quality plan is an organized set of plans, policies, procedures, standards, and practices to achieve consistent, high-quality products and services for customers. It includes all aspects of product development, production, distribution, service delivery, and customer support.
- The quality plan provides a framework for managing quality within an organization. A quality plan should be developed by senior management with input from the entire organization.

The quality plan must include :

- A definition of what constitutes high-quality products or services.
- An explanation of how quality will be measured (quality objectives).
- A description of how quality will be improved through continuous improvement initiatives.
- A description and justification of the organizational structure and processes used to manage quality.
- A summary of the key stakeholders involved in the quality process.
- A description of the roles and responsibilities of each stakeholder group.
- A description of the tools and techniques used to measure and improve quality.
- A process to manage changes
- A description of how the quality plan will be communicated to employees.
- A statement indicating the commitment of the organization to achieving high quality.
- In summary, the Quality Plan is a roadmap to achieve quality.

☞ **The purpose of a Quality Plan**

Quality Plan

GQ. Defines quality expectations.

Focus on meeting customer expectations

- Establishes a mechanism to check compliance with the requirements. (inspection, tests and audits)
- Provides confidence to customers Sometimes customers ask to provide a copy of the Quality Plan for complex products or projects.
- A quality plan is a document, or several documents, that together specify quality standards, practices, resources, specifications, and the sequence of activities relevant to a particular product, service, project, or contract. Quality plans should define:
- Objectives to be attained (for example, characteristics or specifications, uniformity, effectiveness, aesthetics, cycle time, cost, natural resources, utilization, yield, dependability, and so on)
- Steps in the processes that constitute the operating practice or procedures of the organization
- Allocation of responsibilities, authority, and resources during the different phases of the process or project
- Specific documented standards, practices, procedures, and instructions to be applied
- Suitable testing, inspection, examination, and audit programs at appropriate stages
- A documented procedure for changes and modifications to a quality plan as a process is improved
- A method for measuring the achievement of the quality objectives
- Other actions necessary to meet the objectives
- At the highest level, quality goals and plans should be integrated with overall strategic plans of the organization. As organizational objectives and plans are deployed throughout the organization, each function fashions its own best way for contributing to the top-level goals and objectives.
- At lower levels, the quality plan assumes the role of an actionable plan. Such plans may take many different forms depending on the outcome they are to produce. Quality plans may also be represented by more than one type of document to produce a given outcome.

4.14.1 Quality Plan Documentation and Deployment

- Quality plans result from both deployed strategic quality policies (which are linked to organizational strategic plans) and from the specific legal regulations, industry standards, organization policies and procedures, internal guidelines, and good practices needed to meet customers' requirements for products or services.
- Strategic-level quality plans are developed and deployed through the strategic planning process. These broad-based quality plans become the guideline for each function's or department's supporting quality plan. Where appropriate, each function or department may develop and internally deploy operating-level quality plans.
- Operating-level quality plans often are the resulting document(s) from a production scheduling function. As such, this documentation often includes blueprints, a copy of the customer's order, references to applicable standards, practices, procedures, and work instructions, and details on how to produce the specific product or service.
- When the product or service is produced, the planning documents may be augmented by inspection documentation, statistical process control (SPC) charts, and copies of shipping documents and customer-required certifications.

- In the process, the plans are transformed from documents to records. In a fully computerized system, the documents mentioned may well be interactive computer screens accessed at operators' workplaces and control points. These screens, internally, become records when operators, inspectors, shippers, and others make computer entries to the screens.
- A completed set of matrices, developed by a quality function deployment (QFD) process, may fulfill a component of an organization's quality plan. The purpose of QFD is to capture and deploy the customers' needs and requirements throughout the organization.

Documenting the quality plan(s) has multiple uses, such as :

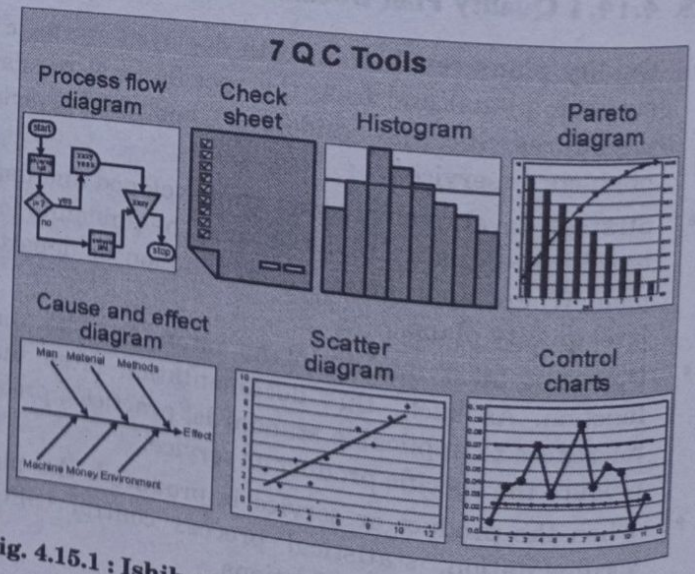
- Ensuring conformance to customer requirements
- Ensuring conformance to external and internal standards and procedures
- Facilitating traceability
- Providing objective evidence
- Furnishing a basis for training
- Together with multiple plans for the organization's products, services, and projects, providing a basis for evaluating the effectiveness and efficiency of the quality management system (QMS).

4.15 QUALITY TOOLS INCLUDING CASE TOOLS

UQ. What are 7 QC Tools and Modern Tools in detail ? **(SPPU - May 18 Endsem)**

UQ. What are Ishikawa's 7 basic tools ? **(SPPU - May 18 Endsem)**

- Dr. Kaoru Ishikawa was first total quality management guru, who has been associated with the development and advocacy of using the seven Quality Control (QC) tools in the organizations for problem solving and process improvements.
- Seven old quality control tools are a set of the QC tools that can be used for improving the performance of the production processes, from the first step of producing a product or service to the last stage of production. So, the general purpose of this paper was to introduce these 7 QC tools.
- This study found that these tools have the significant roles to monitor, obtain, analyze data for detecting and solving the problems of production processes, in order to facilitate the achievement of performance excellence in the organizations.
- There are seven basic quality tools, which can assist an organization for problem solving and process improvements.
- The first guru who proposed seven basic tools was Dr. Kaoru Ishikawa in 1968, by publishing a book entitled "Gemba no QC Shuho" that was concerned managing quality through techniques and practices for Japanese firms.
- It was intended to be applied for "self-study, training of employees by foremen or in QC reading groups in Japan.
- It is in this book that the seven basic quality control tools were first proposed. valuable resource when applying the seven basic tools (Omachonu and Ross, 2004).



(1D3) **Fig. 4.15.1 : Ishikawa7 Basic Tool for Quality Control**

These seven basic quality control tools, which introduced by Dr. Ishikawa, are :

1. Check sheets;
2. Graphs (Trend Analysis);
3. Histograms;
4. Pareto charts;
5. Cause-and-effect diagrams;
6. Scatter diagrams;
7. Control charts.

Fig. 4.15.1 indicates the relationships among these seven tools and their utilizations for the identification and analysis of improvement of quality.

4.16 QUALITY CONTROL AND RELIABILITY OF QUALITY PROCESS

Export of quality products is one of the important determinants of success in international marketing. Inferior quality of the product spoils the image of not only the product but also the nation. So, exporters should be quality conscious and their governments should insist on quality control.

What is Quality Control ?

Quality control is a deliberate and planned activity in order to determine the quality of a product with a view to accepting it as such. If it does not satisfy these requirements, then appropriate remedial measures are taken to correct the process or activity. Quality control is best exercised during the course of production of an article – actually starting with the raw materials, going through the various processing stages and ending up with the final product and paying due attention to packing, storage and transport.

Objectives of Quality Control

The important objectives of the quality control are as follows :

1. To promote and ensure the image of Indian goods exported to other countries.
2. To ensure goods of assured quality alone are sent to the export market.
3. To sustain foreign markets where Indian goods are well received and develop new markets with competitive edge.
4. To instill confidence in the minds of overseas buyers with the assurance provided by the third party guarantee.
5. To adhere strictly to technological requirements of the product accepted by foreign buyers.
6. To ensure sound and safe performance of the products without causing health hazards.
7. To observe conformity of rules and regulations of the importing countries.
8. To maintain proper packaging for the safety of the product during transit.
9. To eliminate the causes of complaints from the foreign buyers and to improve the overall quality of the Indian products.
10. To maximize production by achieving economies of scale.

Quality Standards in quality control

Specification of quality is a must for quality control. Only when quality characteristics are assessed, specified and measured, quality control can be implemented. Generally, buyers give specifications of products which they intend to buy.

However, standards are specified by quality control and inspection institutions such as, The Bureau of Indian Standards and also national standards of the importing country, such as International Standards Organization (ISO) and International Electro-chemical Commission.

The exporters should conform to the specifications stated by these organizations. Exports of products are allowed only when the standards specified are complied with. Apart from this, the Export inspection council has laid down minimum standards for a number of products.

Methods of quality control and inspection

There are two methods of quality control, namely,

(1) In-process quality control and (2) Consignment-wise inspection.

(1) In-process quality control

- In-process quality control covers certain products like paints, and allied products, linoleum, ceramic sanitary ware, printing ink, chrome figments, etc. Under this method, the manufacturers themselves are responsible for producing export consignments conforming to the standard specification.
- Proper control should be exercised at various levels like raw materials control, process control, preservation control and packing control. Adequate controls are taken by periodic inspection and testing of export consignments at random.

(2) Reliability and Quality Control

- Reliability refers to the extent to which the product may perform its functions. A product's useful life span or the expected duration of its performance constitutes its reliability.
- There is a close relationship between reliability and quality control. In some cases, e.g., electronic generating stations, steel plants, etc., reliability is not only desirable but essential. Any downtime of a power plant or steel plant is very expensive.
- Reliability embraces the entire production cycle from product design to product use. Quality control involves checking of standards established during the design stage. But improved workmanship is possible only when production staff are motivated and take pride in workmanship.
- A well planned and organized quality control system starts controlling quality from design stage and continues to work until a trouble free product is supplied to the customer. Customer satisfaction can be achieved only when the customer is happy with the performance, reliability and serviceability of the product.
- Reliability can be measured as the probability of performing without failure, a specified function under given conditions for specified period of time.

4.17 QUALITY MANAGEMENT SYSTEM MODELS

- A quality management system (QMS) is defined as a formalized system that documents processes, procedures, and responsibilities for achieving quality policies and objectives. A QMS helps coordinate and direct an organization's activities to meet customer and regulatory requirements and improve its effectiveness and efficiency on a continuous basis.
- ISO 9001:2015, the international standard specifying requirements for quality management systems, is the most prominent approach to quality management systems. While some use the term "QMS" to describe the ISO 9001 standard or the group of documents detailing the QMS, it actually refers to the entirety of the system. The documents only serve to describe the system.

Benefits of Quality Management Systems

- Implementing a quality management system affects every aspect of an organization's performance. Benefits of a documented quality management system include:
- Meeting the customer's requirements, which helps to instill confidence in the organization, in turn leading to more customers, more sales, and more repeat business
- Meeting the organization's requirements, which ensures compliance with regulations and provision of products and services in the most cost- and resource-efficient manner, creating room for expansion, growth, and profit

These benefits offer additional advantages, including :

(1) Defining, improving, and controlling processes
(2) Reducing waste
(3) Preventing mistakes
(4) Lowering costs
(5) Facilitating and identifying training opportunities
(6) Engaging staff
(7) Setting organization-wide direction
(8) Communicating a readiness to produce consistent results

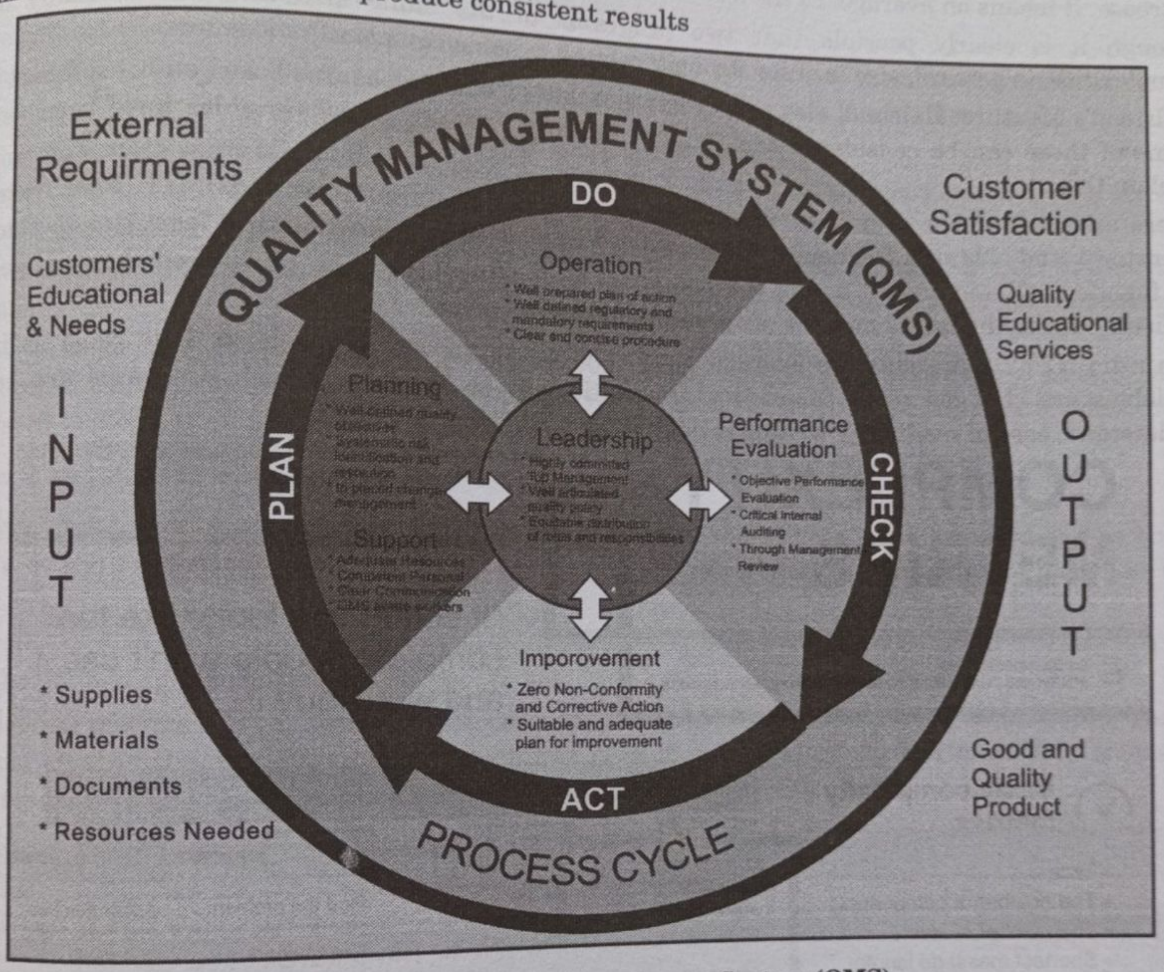


Fig. 4.17.1 : Quality Management System (QMS)

4.18 COMPLEXITY METRICS AND CUSTOMER SATISFACTION

- The productivity, if measured only in terms of lines of code per unit of time, can vary a lot depending on the complexity of the system to be developed.
- A programmer will produce a lesser amount of code for highly complex system programs, as compared to a simple application program. Similarly, complexity has a great impact on the cost of maintaining a program.
- To quantify complexity beyond the fuzzy notion of the ease with which a program can be constructed or comprehended, some metrics to measure the complexity of a program are needed.
- A complexity measure is a cyclomatic complexity in which the complexity of a module is the number of independent cycles in the flow graph of the module. A number of metrics have been proposed for quantifying the complexity of a program, and studies have been done to correlate the complexity with maintenance effort. In this article, we will discuss a few complexity measures. Most of these have been proposed in the context of programs, but they can be applied or adapted for detailed design as well.

- Size Measures: A complexity measure tries to capture the level of difficulty in understanding a module. In other words, it tries to quantify a cognitive aspect of a program. It is well known that, in general, the larger a module, the more difficult it is to comprehend.
- Hence, the size of a module can be taken as a simple measure of the complexity of the module. It can be seen that, on average, as the size of the module increases, the number of decisions in it is likely to increase. It means on average, as the size increases, the cyclomatic complexity also increases.
- Though it is clearly possible that two programs of the same size have substantially different complexities, in general, size is quite strongly related to some complexity measures.
- Halstead's Measure: Halstead also proposed a number of other measures based on his software science. Some of these can be considered complexity measures. A number of variables have been defined to explain this.
- These are n1 (number of unique operators), n2 (number of unique operands), N1 (total frequency of operators), and N2 (total frequency of operands). As any program must have at least two operators : one for function call and one for end of the statement the ratio n1/2 can be considered the relative level of difficulty due to the larger number of operators in the program.
- The ratio N2/n2 represents the average number of times an operand is used. In a program in which variables are changed more frequently, this ratio will be larger. As such programs are harder to understand, ease of reading or writing is defined as:

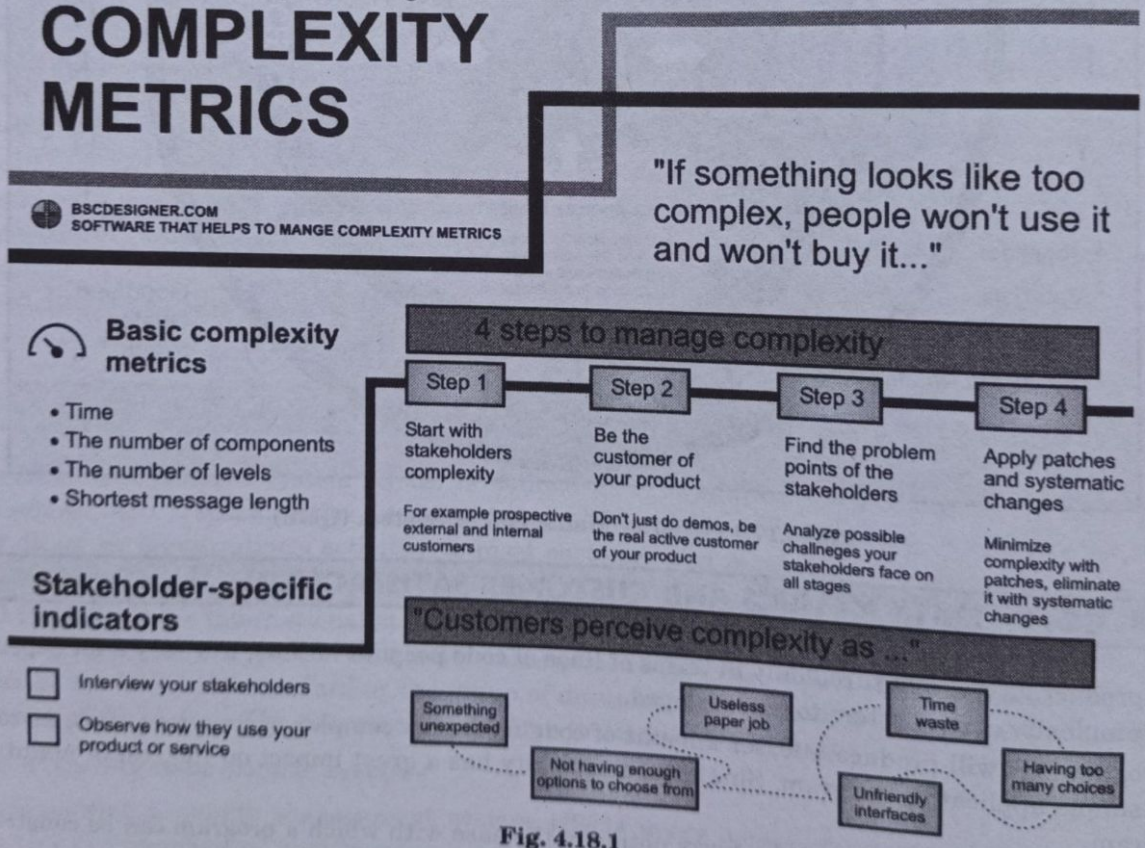


Fig. 4.18.1

- Customer satisfaction metrics reveal crucial data that helps you understand how delighted customers are with your products, services, or the overall business. In other words, customer satisfaction metrics reveal the quality of a customer's relationship with your brand.
- It must be noted that customer metrics reveal more than just numerical scores. They reveal the valuable opinions, attitudes, behavioral patterns, and feedback of your audience.

- Some examples of customer satisfaction KPIs include Customer Effort Score (CES), Net Promoter Score (NPS), Customer Churn Rate (CCR), etc.
- Although each and every metric has an important role to play, you need to ensure they align well with your business goals. For instance, if your goal is to create a simple and easy-to-use product, you can prioritize a metric such as the Customer Effort Score.

GQ. Why Do You Need to Measure Customer Satisfaction ?

- Customer satisfaction is no longer just a buzzword that brands use to attract customers. A great emphasis on customer satisfaction can give your business an edge over your competitors.

Create the Right Culture of Communication & Feedback

- Modern customers want to do business with a brand that values their opinions and is always striving to improve.
- Monitoring customer satisfaction is a great way to keep customers engaged with your brand. It gives the message that your business is always open for communication and cares about the well-being of its customers.
- Moreover, with the help of customer feedback, you can get a good idea about their preferences. For instance, if CSAT surveys reveal that customers are more satisfied watching online video tutorials compared to blogs, then you can focus on creating more such videos.

Forecast Business Growth & Revenue

- Happy and satisfied customers not just buy more but also act as brand ambassadors and recommend your products or services to others. In short, they do the job of your marketing team, and that too for free.
- According to a study by Temkin Group, around 77% of customers would recommend a brand to a friend after having a single positive experience.
- By adopting metrics such as the NPS, you can easily measure the likelihood of new and repeat business. You can identify the proportion of your customer base that is loyal to you and therefore, forecast cash flow, repeat purchases, referrals, and business growth.

4.19 INTERNATIONAL QUALITY STANDARDS – ISO, CMM

UQ. Write in brief ISO-9000 series, CMM, CMMI, Test Maturity Models ? SPPU - Aug. 18 Insem

GQ. What is the ISO 9001 Standard ?

- The ISO 9001 standard is a document that describes all of the requirements needed in order to create and maintain a quality management system as described in ISO 9000. This is a subtle difference between ISO 9000 and ISO 9001 that some fail to recognize.
- So, to explicitly point it out, the difference between the two (ISO 9000 vs 9001) is summarized as the definition of quality management system (ISO 9000) and requirements needed to meet that definition (ISO 9001).
- Both the ISO 9000 and 9001 standards are based on a number of quality management principles including a strong customer focus, the motivation, and implication of top management, the process approach and continual improvement.

The seven quality management principles include the following as described by the ISO :

1. **Customer focus :** Quality management primarily focuses on meeting customer requirements and striving to exceed customer expectations.
2. **Leadership :** Helping leaders to establish unity of purpose and direction at all levels and to create conditions to engage members of the organization in achieving the organization's quality objectives.

3. **Engagement of people :** Obtaining and maintaining (at all levels throughout the organization) competent, empowered, and engaged people to enhance the organization's capability to create and deliver value.
4. **Process approach :** Delivering consistent and predictable results through the use of effective and efficient activities that are understood and managed as interrelated processes that function as a coherent system.
5. **Improvement :** Maintaining an ongoing, organization-wide focus on improvement.
6. **Evidence-based decision making :** Using the analysis and evaluation of data and information in the decision making process to produce desired results.
7. **Relationship management :** Managing the organization's relationships with related parties, such as partners or vendors, for sustained success.

- You can learn more about the quality management principles from the ISO's 2015 publication. Also, a copy of ISO 9001:2015 can be purchased at the ISO online store.

4.19.1 Why ISO 9000 or 9001 ?

- One misconception is that ISO 9000 or 9001 is only for manufactures or large organizations. As a principles-based standard, ISO 9001 can be applied to any organization regardless of what type or size it may be.
- The standard defines the requirements, but it does not dictate the method of application.
- The latest version of the standard has been specifically designed to be more accessible to organizations outside the manufacturing sector.
- As with anything, there are ISO 9000/9001 pros and cons. The application of ISO 9001 when implementing a quality management system can provide the following benefits the organizations:
- Clear understanding of your objectives and new business opportunities. Identifying and addressing the risks associated with your organization. Renewed emphasis on putting your customers first. Meeting the necessary statutory and regulatory requirements. Organizational and process alignment to increase productivity and efficiency.

4.19.2 What is an ISO 9000 Certification ?

- If you are researching the ISO 9000 requirements or how to become ISO 9000 certified, you should really be focused on ISO 9001. You see an organization cannot become ISO 9000 certified.
- First issued in 1987 and last updated in 2015, ISO 9001 is the standard that sets out the criteria for a quality management system and is also the only standard within ISO 9000 that an organization can certify to. Therefore, it is incorrect to say that an organization is ISO 9000 compliant.
- However, a business can be ISO 9001 certified or compliant.
- While an ISO 9001 certification is not regulatory requirement, ISO reports that "over one million companies and organizations in over 170 countries have certified to ISO 9001."

An organization must demonstrate the following in order to be ISO 9001 certified :

- The company follows the guidelines within the ISO 9001 standard;
- The company meets its own requirements;
- The company meets its customer requirements and statutory and regulatory requirements; and
- The company maintains documentation of its performance.
- An ISO 9001 certification can enhance an organization's credibility as it shows customers that the organization's products and services meet quality expectations.

Additionally, there are some instances where an ISO 9001 certification is required or legally mandated for businesses in some industries.

4.19.3 How to become ISO 9000 Certified ?

- The ISO 9001 certification process requires an organization to implement ISO 9001:2015 requirements. Once implemented, an organization must successfully complete registrar's audit to confirm that the organization system meets those requirements.
- The auditor will interview management and staff within the organization to determine whether or not they understand their role and responsibilities in complying with the ISO 9001 standards.
- The auditor will also examine the organization's documentation to validate compliance with the ISO 9001 requirements.
- The auditor will then prepare a detailed report that details the parts of the standard that the organization did not meet.
- The organization will need to agree to correct any problems within a specified time frame. The organization executes remedial activities to ensure that all problems are corrected.
- Once these gaps are addressed and confirmed by the auditor, the organization can then be certified.
- In order to maintain the ISO 9001 certification, the organization must continue with regular surveillance and recertification audits.
- Capability Maturity Model is used as a benchmark to measure the maturity of an organization's software process.
- CMM was developed at the Software engineering institute in the late 80's. It was developed as a result of a study financed by the U.S Air Force as a way to evaluate the work of subcontractors. Later based on the CMM-SW model created in 1991 to assess the maturity of software development, multiple other models are integrated with CMM-I they are

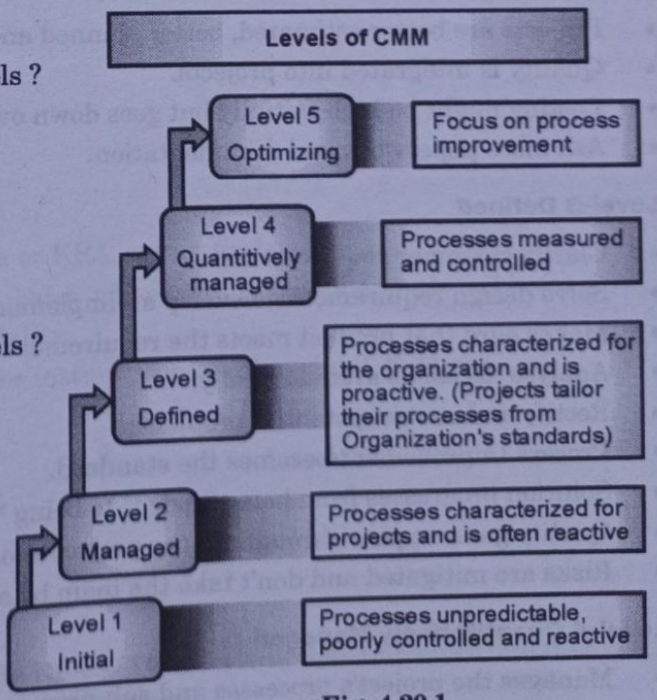
4.20 CAPABILITY MATURITY MODEL (CMM) & CMM LEVELS

In this tutorial, we will learn,

- What is Capability Maturity Model (CMM) Levels ?
- What happens at different levels of CMM?
- How long does it Take to Implement CMM?
- Internal Structure of CMM

Limitations of CMM Models

1. Why Use CMM ?
2. What is Capability Maturity Model (CMM) Levels ?
3. Initial
4. Repeatable/Managed
5. Defined
6. Quantitatively Managed
7. Optimizing



(1E3)**Fig. 4.20.1**

4.20.1 Capability Maturity Model (CMM) With Levels

GQ. What happens at different levels of CMM ?

Levels

- Activities
- Benefits

Level 1 Initial

- At level 1, the process is usually chaotic and ad hoc.
- A capability is characterized on the basis of the individuals and not of the organization.
- Progress not measured.
- Products developed are often schedule and over budget
- Wide variations in the schedule, cost, functionality, and quality targets.
- None. A project is Total Chaos.

Level 2 Managed

- Requirement Management
- Estimate project parameters like cost, schedule, and functionality.
- Measure actual progress.
- Develop plans and process.
- Software project standards are defined.
- Identify and control products, problem reports changes, etc.
- Processes may differ between projects.
- Processes become easier to comprehend.
- Managers and team members spend less time in explaining how things are done and more time in executing it.
- Projects are better estimated, better planned and more flexible.
- Quality is integrated into projects.
- Costing might be high initially but goes down overtime.
- Ask more paperwork and documentation.

Level-3 Defined

- Clarify customer requirements
- Solve design requirements, develop an implementation process.
- Makes sure that product meets the requirements and intended use.
- Analyse decisions systematically.
- Rectify and control potential problems.
- Process Improvement becomes the standard.
- Solution progresses from being "coded" to being "engineered".
- Quality gates appear throughout the project effort with the entire team involved in the process.
- Risks are mitigated and don't take the team by surprise.

Level-4 Quantitatively Managed

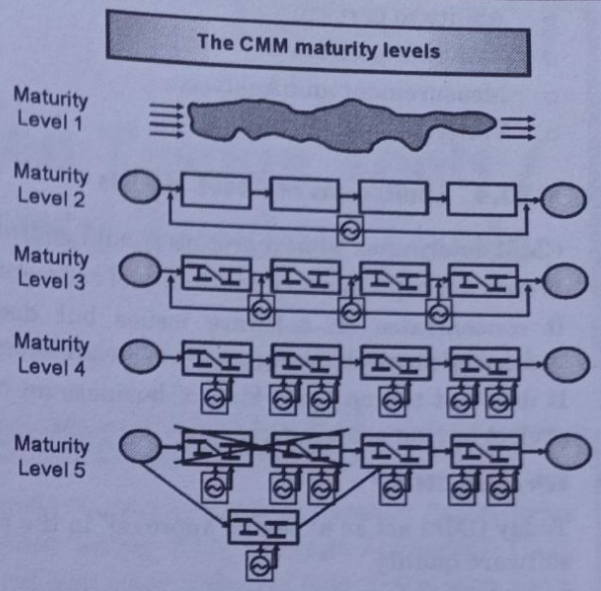
- Manages the project's processes and sub-processes statistically.

- Understand process performance, quantitatively manage the organization's project.
- Optimizes Process Performance across the organization
- Fosters Quantitative Project Management in an organization.

Level-5 Optimizing

- Detect and remove the cause of defects early.
- Identify and deploy new tools and process improvements to meet needs and business objectives.
- Fosters Organizational Innovation and Deployment.
- Gives impetus to Causal Analysis and Resolution.

Following diagram, gives a pictorial representation of what happens at different CMM level :



(1E4)**Fig. 4.20.2**

4.20.2 Capability Maturity Model (CMM) with Maturity Level

How long does it Take to Implement CMM ?

- CMM is the most desirable process to maintain the quality of the product for any software development company, but its implementation takes little longer than what is expected.
- CMM implementation does not occur overnight.
- It's just not merely a "paperwork."
- Typical times for implementation is
- 3-6 months -> for preparation
- 6-12 months -> for implementation
- 3 months -> for assessment preparation
- 12 months ->for each new level

4.20.3 Internal Structure of CMM

- Each level in CMM is defined into key process area or KPA, except for level-1.
- Each KPA defines a cluster of related activities, which when performed collectively achieves a set of goals considered vital for improving software capability
- For different CMM levels, there are set of KPA's, for instance for CMM model-2, KPA are
  - REQM - Requirement Management
  - PP - Project Planning
  - PMC - Project Monitoring and Control
  - SAM - Supplier Agreement Management
  - PPQA - Process and Quality Assurance
  - CM - Configuration Management
- Likewise, for other CMM models, you have specific KPA's. To know whether implementation of a KPA is effective, lasting and repeatable, it is mapped on following basis :
  - Commitment to perform

- Ability to perform
- Activities perform
- Measurement and Analysis
- Verifying implementation

4.20.4 Limitations of CMM Models

1. CMM determines what a process should address instead of how it should be implemented.
2. It does not explain every possibility of software process improvement.
3. It concentrates on software issues but does not consider strategic business planning, adopting technologies, establishing product line and managing human resources.
4. It does not tell on what kind of business an organization should be in CMM will not be useful in the project having a crisis right now

Why Use CMM ?

- Today CMM act as a "seal of approval" in the software industry. It helps in various ways to improve the software quality.
- It guides towards repeatable standard process and hence reduce the learning time on how to get things done.
- Practicing CMM means practicing standard protocol for development, which means it not only helps the team to save time but also gives a clear view of what to do and what to expect.
- The quality activities gel well with the project rather than thought of as a separate event.
- It acts as a commuter between the project and the team.
- CMM efforts are always towards the improvement of the process.

Summary

CMM was first introduced in late 80's in U.S Air Force to evaluate the work of subcontractors. Later on, with improved version, it was implemented to track the quality of the software development system. The entire CMM level is divided into five levels.

1. **Level 1 (Initial) :** Where requirements for the system are usually uncertain, misunderstood and uncontrolled. The process is usually chaotic and ad-hoc.
2. **Level 2 (Managed) :** Estimate project cost, schedule, and functionality. Software standards are defined.
3. **Level 3 (Defined) :** Makes sure that product meets the requirements and intended use.
4. **Level 4 (Quantitatively managed) :** Manages the project's processes and sub-processes statistically.
5. **Level 5 (Maturity) :** Identify and deploy new tools and process improvements to meet needs and business objectives.

...Chapter Ends

❑❑❑

CHAPTER

5

UNIT V

Automation Testing Tools / Performance Testing Tools

University Prescribed Syllabus for the Academic Year 2022-2023

Automation Testing : What is automation testing, Automated Testing Process, Automation Frameworks, Benefits of automation testing, how to choose automation testing tools. Selenium Automation Tools: Selenium's Tool Suite- Selenium IDE, Selenium RC, Selenium Web driver, Selenium Grid. Automation Tools: SoapUI, Robotic Process Automation (RPA), Tosca, Appium.

5.1 WHAT IS TEST AUTOMATION ?

UQ. What is automated testing in software testing? How do we decide what to automate in testing and the scope of automation ? **SPPU - Aug.18(Insem), Oct. 19 (Insem)**

GQ. What is Test Automation ?

(a) Different terms used in automation.

(b) Skills needed for automation.

(c) What to automate.

(d) How to define scope of automation. **SPPU - Aug 18 (Insem)**

5.1.1 Developing Software to Test the Software is called Test Automation

- Test automation can help address several problems. Automation saves time as software can perform test cases faster than human do this can help in operation the tests immediately or unattended. The time thus saved can be used efficiently for analysis engineers to,
- Develop other test cases to realize better reporting; Perform some obscure or dedicated tests like ad hoc difficult; or Perform some extra manual testing.
- The time saved in computerization can also be utilize to expand further test cases, thereby civilizing the reporting of testing. Moreover, the computerized tests can be run overnight, saving the elapsed time for testing, thereby enabling the product to be released frequently.
- **Test computerization can free the test engineers from mundane tasks and make them focus on more original tasks -** If present are too many designed test cases that need to be run yourself and sufficient mechanization does not exist, then the test team may expend most of its time in test implementation.
- **Automated tests can be more reliable -** When an engineer executes an exacting test case many times manually, there is a chance for human error or a bias because of which some of the defect may get missed out.
- As with all machine-oriented activities, automation can be expected to produce more reliable results every time, and eliminates the factors of boredom and fatigue.

5.1.2 Automation Help in Instantaneous Difficult

Automation reduces the time gap between increase and testing as scripts can be execute as soon as the product build is ready.

Automation can be designed in such a way that the tests can be kicked off automatically, after a successful build is over.

- **Automation can protect an association against attrition of test engineers -** Automation can also be used as a information transport tool to train test engineers on the product as it has a depository of dissimilar tests for the product.
- Test computerization opens up opportunity for better operation of global property
- Manual testing require the attendance of test engineers, but automatic tests can be run round the clock, twenty-four hours a day and seven being a week.
- This will also allow team in dissimilar part of the world, in dissimilar time zone, to observe and organize the tests, thus given that round the clock exposure.
- Certain types of difficult cannot be execute without computerization

- Test cases for certain types testing such as reliability testing, stress testing, load and performance testing, cannot be execute exclusive of automation.
- For instance, if we desire to study the performance of a scheme with thousands of user logged in, there is no method one can achieve these tests exclusive of using automatic tools

5.1.3 Automation Makes the Software to Test the Software and Enables the Human Effort to be Spent on Original Testing

- **Automation means end to end, not test completing alone** Automation does not end with developing programs for the test cases. In fact, that is where it starts.
- Automation should believe all behavior such as alternative up the right produce build, choosing the right design, the stage installation, organization the tests, generate the right test data, analyze the results, and filing the defect in the imperfection repository.
- Automation is supposed to have scripts that produce examination data to make the most of coverage of permutation and combinations of inputs and predictable output for product contrast. They are called *test data generator*
- It is not constantly easy to see coming the output for all input situation.
- Even if all input situation are known and the predictable results are met, the error produced by the software and the functioning of the product after such an error may not be predictable.
- The automation script be supposed to be able to map the error patterns dynamically to conclude the result.

5.2 AUTOMATED TESTING PROCESS

A ***Test Case*** is a set of sequential steps to execute a test in use on a set of predefined inputs to produce assured predictable outputs.

5.2.1 There is Two Types of Test cases Automated and Manual

1. A test case (manual or automated) can be represented in many forms.
2. It can be documented when a set of simple steps, or it could be an assertion statement otherwise a set of assertions. An example of an assertion is **"Opening a file, which is already opened should fail."**
3. An assertion statement includes the expected result in the definition itself, as in the above example, and makes it easy for the automation engineer to write the code for the steps and to conclude the correctness of result of the analysis case.
4. Testing involves several phases and several types of testing.
5. This presents an opportunity for the automation code to be reused for different purposes and scenarios.

- The Fig. 5.2.1, **Continuous Test Automation Maturity Model** is a helpful tool to establish the best fit for the maturity of an organization or application within an organization.
- By advertising the kind that best match, give a visual picture of the leading level of maturity. This also is a fast way to decide areas to address to recover the stage of maturity.

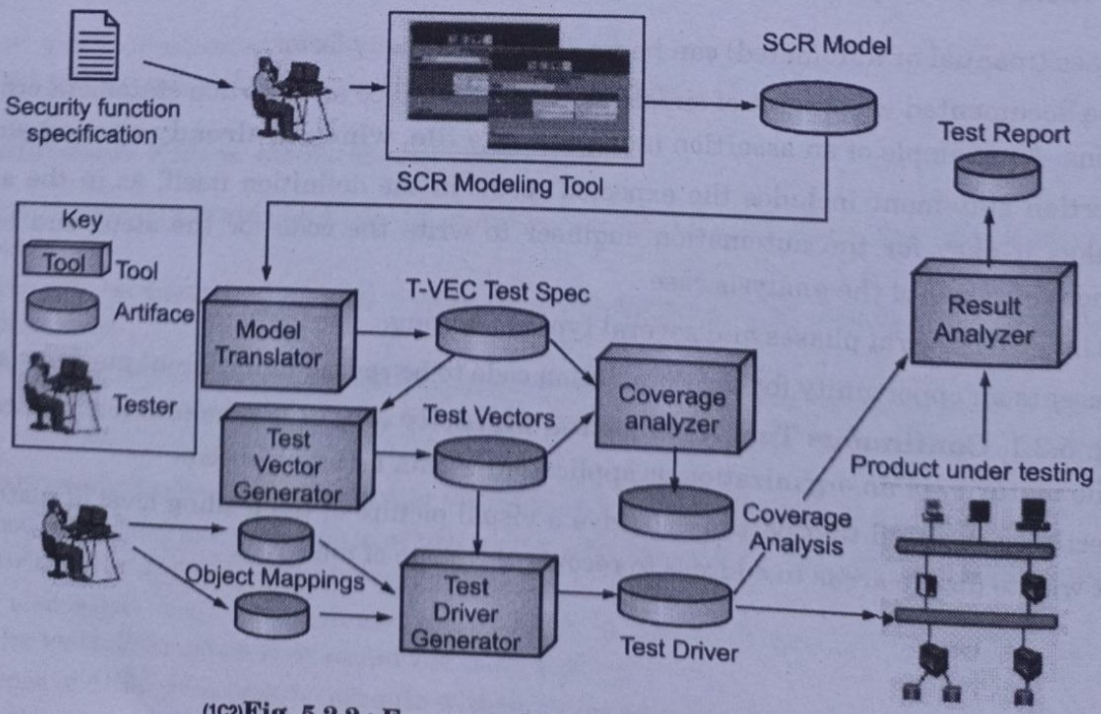
	People	Process	Tech
Chaos 1	☐ Silo team organization ☐ Little knowledge of test automation ☐ Blame, finger-pointing	☐ Testing not part of planning ☐ No test standards ☐ Few automated tests	☐ Missing tools to test performance of applications, pipelines and infrastructure
Continuous integration 2	☐ Limited knowledge of testing, Ad-hoc training ☐ Some Dev/QA co-ordination	☐ Most test other than build tests are manual ☐ Minimal test version management	☐ Version management ☐ Automated build tests ☐ Painful but repeatable releases
Continuous flow 3	☐ Test automation skills and training program ☐ Risk management ☐ Dev/QA joint plan	☐ E2E CI/CD pipeline, test visible ☐ Test/release standards ☐ Test managemet	☐ Most tests automated for app, infra, pipeline ☐ Release metrics use test results
Continuous Feedback 4	☐ Collaboration using shared test metrics ☐ Goals: SLI/As, ☐ Mentors and Guilds	☐ E2E performance trends drive test design ☐ Focus on removing test bottlenecks	☐ Test environment orchestration ☐ Predictive test analytics
Continuous Improvement 5	☐ Experimentation ☐ Integrated Dev/QA ☐ E2E user experience focus	☐ Risk based test design ☐ Automated test creation and test results analysis	☐ E2E value stream test analysis, orchestration and execution ☐ Intelligent test creation

(1C1)**Fig. 5.2.1 : Constant Test Automation Maturity Model**

5.2.2 There are Two Important Dimensions

"What operations have to be tested," and "how the operations have to be tested."

- The how portion of the examination case is called scenarios (shown in Italics in the above table). "What an operation has to do" is a product-specific feature and "how they are to be run" is a framework-specific requirement.
- The automation belief is based on the fact that product operations (such as log in) are repetitive in nature and by automating the basic operations and leaving the different scenarios (how to test) to the framework/test tool, great progress can be made.



(1C2)**Fig. 5.2.2 : Framework for test automation**

- When a locate of test cases is combined and associated with a set of scenarios, they are called "test suite." A Test suite is not anything but a place of test cases that are automatic and scenarios that are associated with the test cases.
- This provides an overview of Model-Based Testing (MBT) and its activities. A categorization of MBT based on dissimilar criterion is also presented. additionally, several problems of MBT are highlighted.
- A review that provide a thorough report of how MBT is efficient in difficult dissimilar excellence attributes of distributed systems such as protection, presentation, dependability, and precision is given.
- In private communication, it would be fairly straightforward to model distributed system relationships in TTM(T-VEC Tabular Modeller) and generate vectors that could test those relationships, provided you had a tested/test environment designed to set up, control, and record your distributed systems environment in the manner of the test vectors that would result from such an approach.

5.3 AUTOMATION FRAMEWORKS

There are different "*Generations of Automation.*" The skills required for automation depends on what generation of automation the company is in or desires to be in the near future.

5.3.1 The Automation of Testing is Mostly Classify into Three Generation

1. First generation: evidence and Playback
2. Second generation: Data-driven
3. Third generation: Action-driven

1. First generation : Record and Playback

- Record and playback avoid the repetitive nature of executing tests. Almost all the test tools accessible in the marketplace have the evidence and playback feature.
- A test engineer *records* the succession of actions by keyboard typeset or mouse click and those record scripts are *played back* later, in the same order as they be record.
- Since a record script can be play back multiple times, it reduces the tedium of the testing function.

2. Second generation : Data-driven

- This method helps in developing test scripts that generates the set of input conditions and corresponding expected output. This enable the tests to be frequent for unlike input and output conditions. The approach takes as much time and effort as the produce.
- However, change to request does not need the automatic test cases to be changed as long as the input conditions and expected output are still valid.

3. Third invention Action motivated

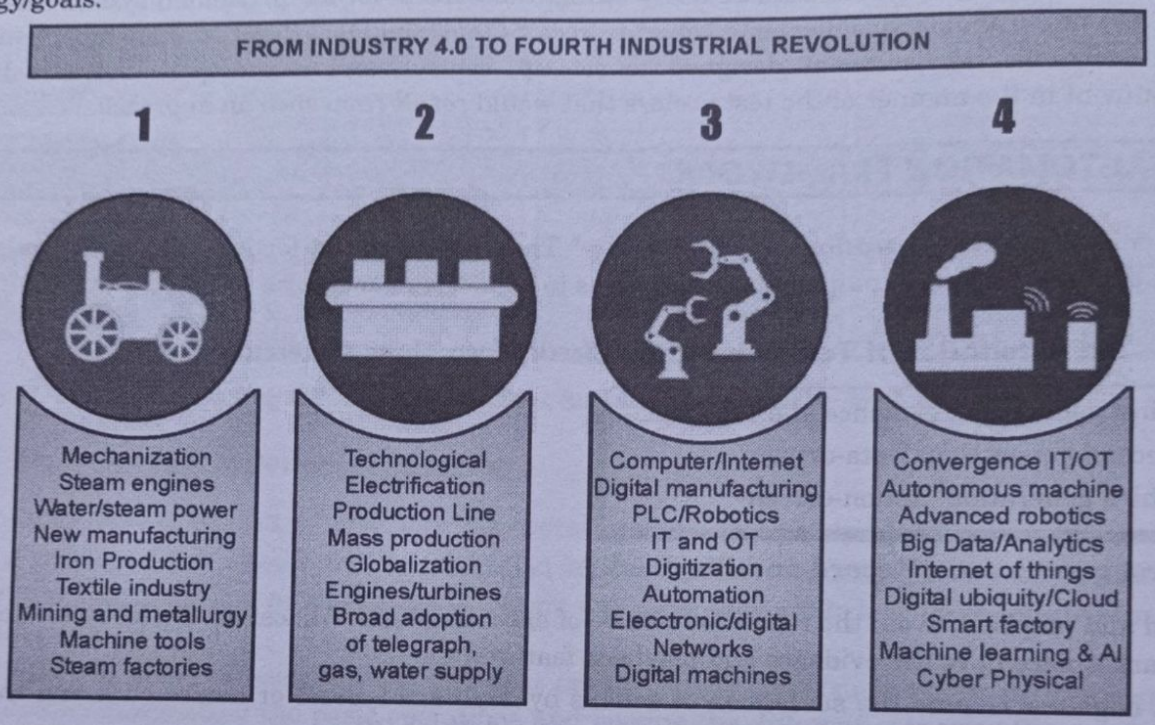
- This technique enables a layman to create automatic tests. Present are no input and expected output situation necessary for organization the tests.
- All events that appear on the request are mechanically tested, based on a general set of control defined for automation.
- The set of actions are represented as objects and those objects are reused. The user wants to specify only the operations (such as log in, download, and so on) and everything else that is needed for those actions are automatically generated.
- The input and output situation are automatically generated and used.

5.3.2 Automation in the Third Generation Involves Two Major Aspects

"Test case automation" and "framework design."

Industry 4.0 is a vision and concept in motion, with reference architectures, standardization and even definitions in flux.

Most Industry 4.0 initiatives are early-stage projects with a limited scope. The majority of digitization and digitalization efforts, in reality, occur in the circumstance of third and even second industrial rebellion technology/goals.



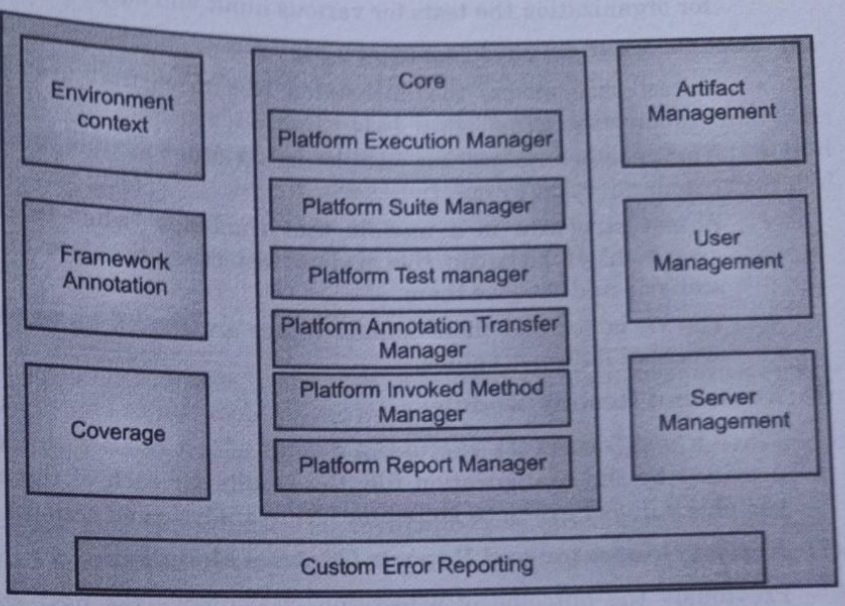
(1C3) **Fig. 5.3.1 : Industry 4.0 - digital alteration of developed in the fourth manufacturing revolution**

- In spirit, the technology creation Industry 4.0 probable control existing data and sufficient additional data sources, counting data from associated assets to increase efficiencies on numerous levels, change existing developed processes, produce end to end in sequence stream across the value chain and understand new forces and business models.
- To **know Industry 4.0,** it is necessary to observe the full value chain which includes supplier and the origins of the resources and mechanism essential for a variety of forms of smart manufacturing, the end-to-end digital supply chain and the final purpose of all developed/production, despite of the number of mediator ladder and company: the ending customer.
- Enabling more straight models of modified production, service, as glowing as purchaser/customer interaction *(including gaining real-time data from actual product usage)* and cutting the inefficiencies, irrelevance and costs of intermediaries in a digital supply chain model, where possible, are some goals of **Industry 4.0** in this customer-centric sense of progressively more difficult clients who value speed, *(cost)* efficiencies and value-added innovative services.
- In the end, it remainders business with the inventive twist of novelty and alteration of business models and process: amplify profit, reduce costs, enhance customer experience, and optimize customer lifetime value and where possible customer loyalty, sells more, and innovates to produce and stay applicable.
- The **Automation Framework Core** is the main module and contains the major structure blocks of the framework. It uses Testing spectators to systematize the test flow and execute the following operation as suitable for each phase of the test implementation :

- Server start up and minimize
- Emma instrumentation, report production, and organization
- Artifact operation and organization
- resident creation and registering
- Report generation

The **Automation Framework API** provides a test pleasant, united advance for confirmation and statement of each service. It includes the following functionality:

- Test friendly covering of the back-end service stubs.
- Test pleasant functions for confirmation and declaration of functions.
- Updates with the relevant release version.
- Encapsulate the complexity of altering all available tests in the case of a remains change.
- Maintain page object classes for Selenium mechanization.



(1C4)**Fig. 5.3.2 : Automation Framework Core**

The **Automation structure Utils** are utility mechanism that provides regularly used functionality, such as distribution SOAP and REST needs to a recognized endpoint or monitor a port to decide whether it's obtainable. Developers can just add the Utils module to their test case to get the functionality they need. Some example utilities would be:

- Axis2 client
- Wire message monitor
- Custom server starts up scenario (Axis2, Tomcat, Qpid)
- Concurrency test scenario

Testing team and developers classically preserve a set of automation tests by means of well-known mechanization tools like Selenium (for UI automation), J Meter, and Soap UI.

The **Automation Framework Tools** module supports by means of these existing scripts in your test cases and allow you to confirm and report the result of the implementation with other test cases handle by the mechanization Framework.

5.3.3 Modules

(1) **External Module :** There are two module that are external module to automation - TCDB and defect DB. To recall, all the test cases, the steps to execute them, and the history of their execution (such as when a exacting test case was run and whether it passed/failed) are stored in the TCDB.

(2) **Defect DB or *defect database* or *defect repository*** contains details of all the defects that are found in various products that are tested in a particular organization.
It contains defects and all the related information (when the defect was found, to whom it is assigned, what is the current status, the type of defect, its impact, and so on).

(3) **Scenario and Configuration File Modules :** As we have seen in earlier sections, scenarios **are nothing but information on "how to execute a particular test case."**

- **A *configuration file* contain a set of variables that are used in automation.** The variables could be for the test framework or for other module in automation such as tools and metrics or for the examination suite or for a set of test gear or for a exacting test case.
- A configuration file is important for organization the test cases for various execution conditions and for organization the tests for various input and output situation and states.

(4) Test Cases and Test Framework Modules

- A test case means the automated test cases that are taken from TCDB and executed by the framework.
- Test case is an object for completing for other module in the planning and does not symbolize any communication by itself.
- A test structure is a module that combines "what to perform" and "how they contain to be executed." It picks up the explicit test cases that are automatic from TCDB and picks up the scenario and execute them.
- The variables and their defined values are chosen up by the test structure and the test cases are executed for those values.

(5) Tools and Results Modules

When a test framework executes a deposit of test cases with a set of scenarios for the different values provided by the configuration file, the results for each of the analysis case along with scenarios and variable values have to be stored for future analysis and action.

(6) Report Generator and Reports / Metrics Modules

Previously the outcome of a test run is available; the next step is to arrange the test reports and metrics. Preparing reports is a complex and time-consuming effort and hence it should be part of the automation design.

5.4 BENEFITS OF AUTOMATION TESTING

The precise supplies can vary from product to product, from situation to situation, from time to time.

5.4.1 We Present below Some Generic tips for Identifying the Scope for Automation

1. **Identifying the Types** of Testing Amenable to Automation
2. **Certain types of tests** automatically lend themselves to automation.
3. **Stress, reliability, scalability, and performance testing**
4. **These types of testing** require the test gear to be run from a large numeral of different machines for an extended period of time, such as 24 hours, 48 hours, and so on.
5. **It is just not likely** to have hundreds of users trying out the product day in and day out
6. **Test cases** belonging to these testing types become the first candidates for automation.
7. **Regression Test :** are repetitive in nature. These test cases are executed multiple times during the product growth phases. Given the tedious nature of the test cases, mechanization will save significant time and effort in the long run.
8. **Functional tests :** These kinds of tests may require a complex set up and thus require specialized skill, which may not be available on an continuing basis.
9. **Automate these once**, with the specialist ability sets, can allow with less expert natives to run these tests on an continuing basis.
10. **This provides** an opening to automate test cases and execute them multiple times during release cycles.

5.4.2 Automating Areas Less Prone to Change

- Automation should consider those areas where requirements go through lesser or no changes. Normally change in requirements cause scenarios and new features to be impacted, not the basic functionality of the product.
- While automating, such basic functionality of the product has to be considered first, so that they be able to be used for "*regression test bed*" and "*daily builds and smoke test.*"
- The non-user interface portions of the product can be automated first. While automating functions involving user interfaces-and non-user interface-oriented ("backend") elements, clear demarcation and "plug ability" have to be provided so that they can be executed together as well as executed independently.
- This enable the non GUI portions of the mechanization to be reuse even when GUI goes during change.

5.4.3 Automate Tests that Pertain to Standards

- Automating for standards provides a dual advantage. Test suites developed for standards are not only used for product testing but can also be sold as test tools for the market.
- A large number of tools obtainable in the commercial market were internally developed for in-house usage.
- Hence, automating for standards creates new opportunities for them to be sold as commercial tools.
- The certification suites are executed every time by the supporting organization before the release of software and hardware. This is called "*certification testing*" and requires perfectly compliant results every time the tests are executed.

5.5 HOW TO CHOOSE AUTOMATION TESTING TOOLS

5.5.1 Selecting the Test Tool is an Important Aspect of Test Automation for Several Reasons as given below

1. **Free tools are *not well supported and get phased out* soon.** It will be extremely dangerous to see a release stalled because of a problem in a test tool.
2. **Developing in-house tools *takes time*.** Even though in-house tools can be less expensive and can meet needs better, they are often developed by the personal interest shown by a few engineers
3. **Test tools sold by vendors are expensive.** In absolute dollar terms, the standard test automation tools in the marketplace are expensive.
4. **Test tools require strong training.** Test automation cannot be successful unless the people using the tools are properly trained.
5. **Such training usually involves** getting familiar with the scripting languages that come with the tool, customizing the tool for use, and adding extensions or plug-ins for the tool.
6. **Test tools generally** do not meet all the requirements for automation. Since tools are meant to be generic, they may not fully satisfy the needs of a particular customer. That is why customization and extensibility become key issues.
7. **Not all test tools run on all platforms**. To amortize the costs of automation, the tools and the automated tests should be reusable on all the platforms on which the product under test runs. Portability of the tool and the scripts to multiple platforms is therefore a key factor in deciding the test automation tool.

5.5.2 Criteria for Selecting Test Tools

UQ. What are the selection criteria of automatic difficult tool?

SPPU - Nov./Dec.18(Endsem), Oct. 19(Insem)

The categories are:

1. Meeting requirements	2. Technology expectations
3. Training/skills	4. Management aspects

1. Meeting Requirements

- Firstly, there are plenty of tools obtainable in the marketplace but rarely do they meet *all* the requirements of a given product or a given organization. Evaluating different tools for different requirements involves important effort, money, and time.
- Given of the plethora of alternative accessible (with each choice meeting some part of the requirement), huge delay is concerned in select and implanting test tools.

2. Technology expectations

- Firstly, test tools in universal may not agree to test developers to extend/modify the functionality of the structure. Therefore extending the functionality requires going back to the implement dealer and involves further cost and attempt.
- Test tools may not provide the same amount of SDK or exported interfaces as provided by the products. Very few tools obtainable in the marketplace supply source code for extend functionality or fixing a number of problems. Extensibility and customization are significant potential of a test tool.
- Secondly, a good quality number of test apparatus need their libraries to be connected with creation binaries. When these libraries are connected by means of the source code of the product, it is called *instrumented code.*
- This causes portions of the testing be repetitive after those libraries are distant, as the results of confident types of difficult will be dissimilar and better when those libraries are impassive.

3. Training skills

- While test tools require plenty of training, very few vendors provide the training to the required level. Organization level teaching is wanted to organize the test tools, as the user of the test suite are not only the test team but also the progress team and other area like configuration management.
- Test tools suppose the user to study new language / scripts and may not use standard language / script.
- This increases skill necessities for mechanization and increase the need for a learning curve inside the organization.

4. Management aspects

- A test tool increases the system requirement and requires the hardware and software to be upgraded. This increases the cost of the already-expensive test tool.
- When selecting the test tool, it is important to note the system requirements, and the cost involved in upgrading the software and hardware needs to be included with the cost of the tool.
- Let's talk more about evaluating test automation tools for Object Recognition for testing web applications. You'll need to evaluate which browsers are supported by each tool one by one. We normally check the top 3 popular browsers – Firefox, IE (Edge), and Chrome. Each uses a different rendering engine – Gecko, Trident, and Webkit. Because of this, automation tools need separate methods while working with them.

- The Table 5.5.1 only gives a very basic and a quick glance of the capability for Object Recognition for desktop and web, but you may require to include mobile test automation as well. For the full evaluation, we are more specific and expand into more details that are more contextual and targeted toward the software under test.

Table 5.5.1 : Glance of the capability for Object Recognition for desktop and web

	Desktop Application		Web based Application	
	Rating	Comment	Rate	Comment
Visual Studio Coded UI	Good	Can recognize most controls, except some special tabs.	Good	Can recognize almost all controls, except some special objects, for example a map. The recording feature is good. It supports IE and Firefox, but supports IE better than Firefox.
HP QTP/UFT	Poor	Can recognize few controls	Bad	Cannot recognize controls needed for meaningful scripts
TestComplete	Average	Can recognize some controls. More than QTP but less than Coded UI.	Poor	Can recognize some controls, Many use coordinate positions
Selenium (Webdriver)	Not supported	Does not support	Excellent	With Selenium, almost all controls can be recognized
Ranorex	Average	Can recognize few controls.	Poor	Can recognize some controls. Many use coordinate positions
Silktest	Poor	Can recognize few controls	Good	Same as Coded UI, recognized most controls, The recording feature is good with IE

5.6 INTRODUCTION TO SELENIUM TESTING ?

SPPU - Aug 19 (Insem), May 18 (Endsem)

UQ. What Is Selenium Tool ?

- **Selenium** is a free Open source functional Testing tool used for testing web applications on multiple browsers and multiple operating systems (Platforms). It is used for Functional and Regression Testing. Testing done by the selenium tool is usually referred to as Selenium Testing.
- SQA includes all software growth procedures starting from important necessities to coding until issue. Its prime goal is to ensure quality.

5.6.1 Brief History of the Selenium Project

SPPU - Aug 2017(Insem), May 18(endsem)
SPPU - Nov./Dec.19(Endsem)

UQ. Brief History of Selenium Project ?

UQ. What is Selenium Project ?

- Selenium (the testing framework, not the mineral you get from eating clams) came from? Here's a short history of the technology, from its origins more than a decade ago as a proprietary tool through the present era of Webdriver.

- Selenium originated in elder days - by which I mean 2004 - as a tool for testing web applications. It was developed by Jason Huggins, a programmer at Thought-Works.
- That Selenium originated at Thought-Works is interesting. While no one in 2004 was talking about Agile infrastructure, Thought-Works was the place where Martin Fowler made his career. Fowler went on to become one of the major thought leaders behind the migration to micro services.
- While Fowler can't take credit for Selenium, it seems fitting that the tool, which is an important part of automated testing for Develops-inspired workflows today, originated in the same place from which the Agile infrastructure revolution later emerged.
  - **Open Source Selenium :** At first, Selenium was used only internally by Thought Works employees. But that changed by the end of 2004, when the tool was open-sourced. I don't know exactly when Selenium became an open source tool, since the earliest emails relating to the open-sourcing do not seem to exist anymore (at least not publicly). But it was apparently no later than late November 2004.
  - That's when the first extant Selenium development emails were being exchanged, and people were talking about checking out the Selenium code via Subversion. Selenium at this point remained an imperfect tool. It suffered from some bugs that affected testing for certain browser environments.
  - And thanks to the inefficiencies of the waterfall-style software development practices of the time, bug fixes were slow to reach users. As a Selenium user noted on November 29, 2004.Last but not least, the move placed Selenium within the rapidly growing stack of open source applications at the time.
  - The early 2000s were the era when companies like Red Hat were showing that Linux and other software that was given away for free could have huge commercial value.
  - It was also when open source web browsers, namely Mozilla Firefox, and word processors, such as Open Office, were giving closed-source tools a run for their money. And open source web browsers like Apache HTTP had already held a majority of market share for years. Against this backdrop, open-sourcing Selenium only made sense.
  - Selenium Grows Up : Within a year of its release as an open source tool, Selenium had evolved significantly. By October 2005, developers were dreaming up ambitious "grand plans" for the tool. They envisioned adding sophisticated new features that would extend Selenium beyond its original mission as a basic web app testing tool. Those included things like support for testing framed applications and cross-platform testing. The evolution of Selenium during this period was helped by the fact that Jason Huggins, the original Selenium developer, moved in 2007 to Google, where he was able to continue work on the tool.
  - Selenium Meets Web-driver Simon Stewart developed another testing tool for web apps called WebDriver. WebDriver's debut signaled a desire for features that were not available in Selenium. And for a short time, the two tools competed.
  - But in 2011, the projects were merged to form one web testing tool to rule them all. The combination of Selenium and Web Driver became Selenium 2.0, which debuted in July 2011. The new release paired the Web Driver APIs that are familiar to Selenium users today with the original Selenium feature set. Jason Huggins & Simon StewartS
  - Selenium Present, Selenium Future : The demands of automated testing continue to change. It's a safe bet that Selenium will, too.
  - One important trend that is likely to shape Selenium development going forward is the demand for ever-more efficient automated testing. The Selenium ecosystem has offered some automation options for a while thanks to Selenium Grid and other tools.

- But as the movement increases pressure on development teams to test and deliver software even faster than they already do, techniques for speeding tests, such as by offloading them to the cloud and running them in parallel, will remain key.
- Shift-left testing has also become an important part of the automated testing conversation. Selenium is already well suited for shift-left testing, which refers to the practice of performing tests earlier in the development cycle, in order to identify bugs before they slow development. But Selenium users have to choose to take advantage of Selenium in the right way for this purpose.
- Optimizing Selenium today is easier thanks to a rich ecosystem of plugins and integrations that simplify the task of working Selenium into the software delivery pipeline.
- Since development workflows are now more complex than they have ever been, and will probably grow yet more complex over time, the ecosystem surrounding Selenium is poised to remain essential in helping Selenium to remain relevant for modern application testing.

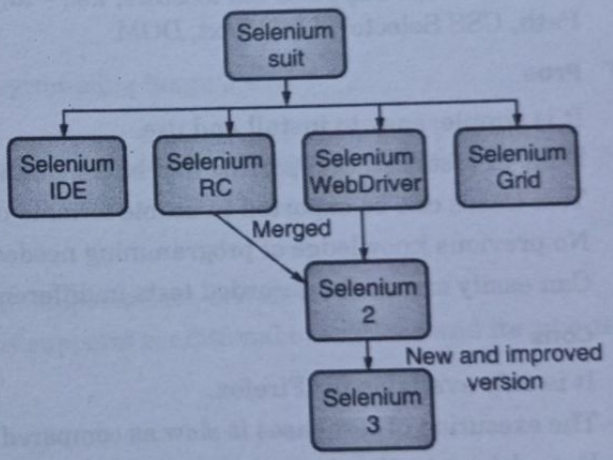
5.7 SELENIUM'S TOOL SUITE

UQ. What is Selenium's IDE explain in detail. SPPU - Oct. 18 (Insem)

UQ. Define Selenium's Tool Suite. List and explain Core Components ? SPPU - Oct. 19 (Insem)

5.7.1 Selenium IDE

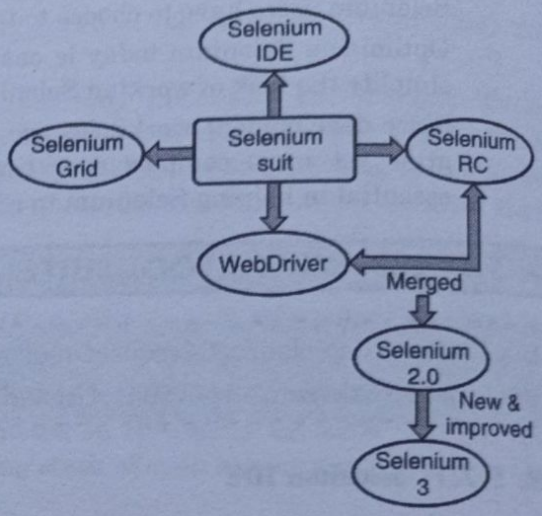
- Selenium IDE (Integrated Development Environment) is an open source web automation testing tool under the Selenium Suite.
- Unlike Selenium WebDriver and RC, it does not require any programming logic to write its test scripts rather you can simply record your interactions with the browser to create test cases. Subsequently, you can use the playback option to re-run the test cases.
- Selenium Integrated Development Environment (IDE) is the simplest framework in the Selenium suite. It is a browser plugin to record and playback the operations performed on the browser.



(101)**Fig. 5.7.1 : Selenium Suite Tree Diagram**

- Selenium IDE plugins are available for Chrome and Firefox browsers. It doesn't support the programming features. Selenium is the language which is used to write test scripts in Selenium IDE.
- As a Firefox plugin, Selenium Integrated Development Environment (IDE) can be used to create a test script prototype quickly and easily. It can record human testers' actions as a script while the tester runs the test case manually.
- Selenium IDE is a rapid prototyping tool for building test scripts within very less amount of time.
- It allows you to record, edit and debug the test case by providing the very simple to use components. This tool will be most helpful for beginners to learn the commands used by selenium while recording the test case. Although it was available as Firefox addon for a long time, it also available on chrome recently.
- With this tool, you can easily record the test case and able to play back any number of time whenever you will require.

- You can easily export the recorded scripts as reusable scripts in one of any programming languages that support.
- Although it was very simple and easy to use tool for beginners, it's having some limitations. It only allows you to record and playback very simple test cases.
- It will not possible to test dynamic websites or web applications. It's neither be scripted using any programming logic not support data-driven testing.
- The recorded test script can be executed at a later point in time for the regression test automatically. This tool can access the browser's DOM elements with the use of JavaScript. It also provides a flexible interface for testers to create or update test cases.
- Thought Works Company introduces selenium IDE in 2006 andimplemented in the Firefox browser, which provides record and playback functionality to the test scripts.
- Selenium-IDE is the simplest tool of Selenium community. Selenium-IDE allows software testers to export recorded scripts in many languages like HTML, Java, Ruby, PHP, Python, C#, and Test-NG.
- Selenium-IDE supports six locators, i.e., - Id, Name, X Path, CSS Selector, Link Text, DOM.



(1D2)**Fig. 5.7.2 : Selenium Suite**

☞ **Pros**

1. It is simple, easy to install and use.
2. Built-in test results reporting and help modules
3. Test Cases can be exported to usable formats in the Selenium RC and WebDriver
4. No previous knowledge of programming needed, though basic knowledge of HTML and DOM required.
5. Can easily export the recorded tests in different programming languages such as Ruby, Python, etc.

☞ **Cons**

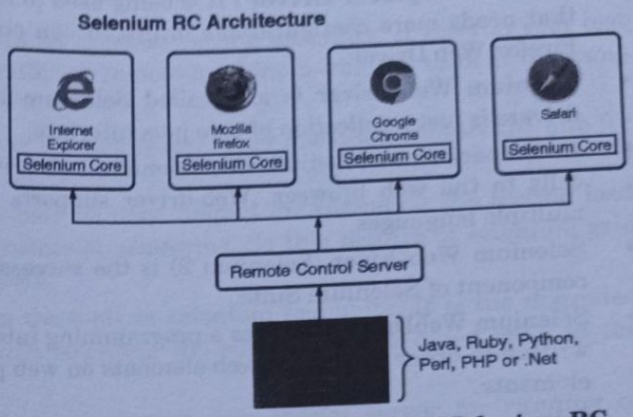
1. It is only available for Firefox.
2. The execution of test cases is slow as compared to RC and WebDriver.
3. Data-driven testing is not supported.
4. It is not able to test dynamic web applications.

5.7.2 Selenium RC

UQ. What is Selenium's RC explain in detail. **SPPU - May 18(Endsem)**

- Selenium RC is the main feature in the Selenium. A tester can use it to simulate user actions such as input data, submit a form, and click a button in web browsers.
- Selenium RC was the first tool used on selenium project. It was the core application written in Java as a programming language.
- This tool will accept commands for the browser via HTTP request. It consists of two components which are selenium RC server and RC client.
- Where RC server will communicate with HTTP/GET/POST request while the RC client will include programming codes.

- You will be able to write an automated test in most of the programming languages such as Java, JavaScript, Ruby, PHP, Python, and Perl and C #.
- Selenium RC is the first open-source tooling in selenium community which is introduced by Thought Works Company in 2004.
- Selenium RC doesn't have a record and playback features. Selenium RC cannot execute test script with selenium server.
- Selenium-RC supports multiple languages (java, C++, python), multiple operating systems (Windows, Linux), and multiple browsers (internet explorer, Google Chrome).
- Although selenium RC was the main project for a long time until the selenium 2 was released. It was officially deprecated when selenium version 2 was released. It will support mostly in maintenance mode that will provide some feature that is not available in Selenium 2.
- When a web browser is being loaded in a test script, it injects a suite of JavaScript (JS) into it. Then use those JavaScript's programming to interact with the different web browsers.



(1D3)**Fig. 5.7.3 : Core Components of Selenium RC**

☞ Usage of Selenium RC

1. Tester writes a test case script with the supported programming language API.
2. The test script sends a command to the RC server.
3. RC server receives these commands and triggers selenium core from executing the commands and interacting with the browser page web elements.

☞ Pros

1. It supports cross-browser testing.
2. It supports data-driven testing.
3. Execution speed is more as compared to IDE.
4. It supports conditional operations and iterations.

☞ Cons

1. Slower execution speed as compared to Web-Driver.
2. Browser interaction is less realistic.
3. Programming knowledge required.

5.7.3 Selenium Web-Driver

SPPU - May 19(End sem)

UQ. What is Selenium's Web-Driver explain in detail.

- Web-Driver is the new feature added in the Selenium 2. It aimed to deliver an easy and helpful programming interface to resolve the limitations of Selenium RC programming API.
- Different from RC, Web-Driver uses browser native support to interact with the web pages.
- So different browsers have different web driver libraries and different features too. All these are decided by the web browser that runs the test cases.
- The implementation of Web-Driver is much more related to the web browser.

So, are the following Web-Driver Drivers ?

- **HttpUnit Driver :** This is one of the fastest and reliable Web-Driver implementations. Based on the HttpUnit, it can run across Linux, Windows, and Mac because of its pure java implementation.
- **Firefox Driver :** It is easy to configure and use. It is being used to run the test scripts in the Firefox web browser and does not require extra configuration to use.
- **Chrome Driver :** It is being used to run a test script on the Google Chrome web browser that needs more configurations to use.
- **Internet Explorer Driver :** It is being used to run the test script in the Internet Explorer web browser that needs more configurations to use. It can only run in Windows OS, slower than the Chrome and Firefox Web Driver.
- Selenium Web-Driver is also called Selenium-2, and Google introduced it in 2008. Selenium Web-drivers is just a collection of core java interface.
- In comparison to Selenium RC, Selenium web driver is more powerful and faster tool because it directly calls to the web browser. Web-driver supports multiple browsers, multiple operating systems, and multiple languages.
- Selenium WebDriver (Selenium 2) is the successor to Selenium RC and is by far the most important component of Selenium Suite.
- Selenium WebDriver provides a programming interface to create and execute test cases. Test scripts are written in order to identify web elements on web pages and then desired actions are performed on those elements.
- Selenium WebDriver performs much faster as compared to Selenium RC because it makes direct calls to the web browsers. RC on the other hand needs an RC server to interact with the web browser.
- Since, WebDriver directly calls the methods of different browsers hence we have separate driver for each browser. Some of the most widely used web drivers include:
  - Mozilla Firefox Driver (Gecko Driver)
  - Google Chrome Driver
  - Internet Explorer Driver
  - Opera Driver
  - Safari Driver
  - HTML Unit Driver (a special headless driver)

☞ **Pros**

1. No separate components such as the RC server are needed.
2. Execution time is faster as compared to Web-Driver and RC.
3. It supports testing on different platforms such as Android, iOS, Windows, Mac, and Linux.

☞ **Cons**

1. No mechanism to track runtime messages.
2. Image testing is not available.
3. Prior knowledge of programming required.

5.7.4 Selenium Grid

UQ. What is Selenium's Grid explain in detail. **SPPU - May 18(End sem)**

- With the Selenium Grid feature, test scripts can run on multiple machines at the same time, which reduces the total test scripts run time.

- Thus helps to find the bug more quickly because the test cases run more quickly. This is suitable for a large application with too many test scripts to run.
- You can also choose to run test scripts on different web browsers and on different machines. You can configure the browser version, Operating System, and machine to run the test case by using the Selenium RC capabilities.
- Selenium grid is the part of selenium version1 that combined with selenium RC to scale for large test suit and able to run tests on remote machines.
- We can execute multiple test cases at the same time on different remote machines. If you run your test cases on multiple environments, you will use the different remote machine to run the tests at the same time.
- While testing with selenium grid, one server acts as the hub where other machines contact the hub for obtaining browser access.
- This ability to run test cases on remote machine may be most helpful for spreading the testing load across several machines having different environments or platforms. So this feature of selenium grid will help you to speed up your test automation process.
- Selenium web driver is the latest addition among the tools in selenium tool suite. It is the upgraded version of Selenium RC. So it is much faster than selenium RC since it makes direct calls to the browsers.
- Unlike the selenium RC, selenium webdriver does not require any special server for running or executing test cases.
- Selenium webdriver supports all of the main browsers such as Mozilla Firefox, Google Chrome and Internet Explorer and Safari. It allows you to write test scripts on different programming languages such as Java, C#, Ruby, Python, Perl etc.
- Although it overcomes all the limitations of Selenium RC, generating a detailed test report is not possible with selenium webdriver yet.
- Selenium Grid is also an important tool of Selenium Suite, which allows us to run our tests script on different machines against different browsers simultaneously.
- Selenium Grid proceeds from the Hub-Node Architecture to achieve parallel execution of test scripts.
- Selenium Grid is divided into two parts:
  - **Grid-1 :** Grid-1 introduced by Thought works company in 2004.
  - **Grid-2 :** Google Company introduced it in 2008.

☞ **Pros**

1. Selenium Grid offers tools needed to diagnose the failures and rebuild a similar environment for the new test execution.
2. Selenium Grid saves time extremely as it uses the Hub-Node design.
3. It supports the simultaneous execution of test cases in multiple browsers and environments.

☞ **Cons**

1. The code executes only on the local machines where the test cases are launched.
2. Considerable efforts and time are must for the initial operation of parallel testing.
3. The remote machine only receives the browser control commands.

5.8 TEST DESIGN CONSIDERATIONS

UQ. What are Selenium's Test Design conditions in detail. **SPPU - May 18(Endsem)**

- The information we provide in this chapter is useful for newcomers and experienced veterans in the field of automated testing.
- This article describes the most common types of automated testing and also describes "design patterns" that can enhance the maintainability and scalability of your automated test suite.
- Experienced automated test engineers who have not yet used these technologies will be more interested in these technologies.

Test type

- What parts of the application should you test? It depends on the various influencing factors of your project: user expectations, time limit, priorities set by the project manager, etc.
- However, once the project boundary is defined, as a test engineer, you must make a decision about what to test.
- In order to classify the type of testing of web applications, we have created some terms here. These terms do not imply standards, but these concepts are typical for web application testing.

Test static content

- Static content testing is the simplest test used to verify the existence of static, unchanged UI elements. E.g. :
  - Every page has its expected page title? This can be used to verify that the link points to an expected page.
  - Does the application's homepage contain an image that should be at the top of the page?
  - Does each page of the website include a footer area to display the company's contact information, privacy policy, and trademark information?
  - Is the <h1> tag used for the title text of each page? Does each page have the correct header text?
- You may or may not need to perform automated tests on the content of the page. If your web content is not easily affected, it is sufficient to test the content manually. If, for example, the location of your application files is moved, content testing is very valuable.

Test link

- A common error on Web sites is invalid links or links to invalid pages.
- Link testing involves clicking on individual links and verifying that the intended page exists. If the static link does not change often, manual testing is sufficient.
- However, if your web designer frequently changes links, or files are redirected from time to time, link testing should be automated.

Function test

- In your application, you need to test specific functions of the application, require some type of user input, and return a certain type of result.
- Usually a functional test will involve multiple pages, a form-based input page, which contains several input fields, submit "and" cancel "operations, and one or more response pages.
- User input can be through text input fields, check boxes, Drop-down list, or any other browser supported input.
- Functional testing is usually the most complex type of test that requires automated testing, but it is also usually the most important.

- Typical tests are logging in, registering website accounts, user account operations, account setting changes, complex data retrieval operations, etc.
- Functional testing usually corresponds to your application's description of application characteristics or designed usage scenarios.

Test dynamic elements

- Usually a web page element has a unique identifier, which is used to uniquely locate the element in the web page. Usually, the unique identifier is implemented with the HTML tag's 'id' attribute or 'name' attribute.
- These identifiers can be a static, i.e. immutable, string constant. They can also be dynamic production values, which change on every page instance.
- For example, some web servers may name the displayed file as doc3861 on a page instance, and display it as doc6148 on other page strengths, depending on the 'document' the user is retrieving.
- The test script that verifies the existence of the file may not be able to locate the file with the same ID.
- Normally, dynamic elements with varying identifiers exist on the results page based on user actions, however, obviously this depends on the web application.
  Below is an example.

```
<input id="addForm:_ID74:_ID75:0:_ID79:0:    checkBox" type="checkbox" value="true"/>
```

This is an HTML marked checkbox,

Its ID (addForm: _ID74: _ID75: 0: _ID79: 0: checkBox) is a dynamically generated value. The next time this page is opened, the ID of the check box may be a different value.

Ajax test

- Ajax is a technology that supports dynamically changing user interface elements. Page elements can be changed dynamically, but do not require the browser to reload the page, such as animations, RSS feeds, other real-time data updates, etc.
- Ajax has countless ways to update elements on a webpage. But the easiest way to understand AJAX is to think of it.
- In Ajax-driven applications, data can be retrieved from the application server and then displayed on the page without reloading the entire page. Only a small part of the page, or only the element itself is reloaded.
- When to use assert command and when to use verify command? It depends on you. The difference is what you want the test program to do when the check fails.
- Do you want the test to terminate, or do you want to continue and simply log the test failure?
- This needs to be weighed. If you use assertions, the test will stop when the check fails, and no subsequent checks will be run.
- Sometimes, perhaps often, this is what you want. If the test fails, you will immediately know that the test failed.
- Test engines such as TestNG and JUnit provide plugins commonly used when developing test scripts, which can easily mark those tests as failed tests.

Advantages

1. You can see directly whether the inspection passed.

☞ **Disadvantages**

1. When the inspection fails, subsequent inspections will not be performed and the result status of those inspections cannot be collected.
2. In contrast, the verify command will not terminate the test. If your test only uses verification, you can be assured that-assuming no unexpected exceptions-the test will be executed regardless of whether a defect is found.

☞ **Cons**

- You have to do more work to check your test results. In other words, you will not get feedback from TestNG and JUnit. You will need to view the results in the printout console or log file.
- Every time you run the test, you need to spend time to check the output. If you are running hundreds of tests, each has its own log, which will take time. Getting feedback in time is more appropriate, so assertions are often used more often than verification.

☞ **Trade-offs**

- assertTextPresent, assertElementPresent and assertText
- You should now be familiar with these commands and the mechanisms for using them. If not, please refer to the relevant chapter. When building your test, you need to decide
- Only check the text on the page? (Verify/assertTextPresent)
- Only check if there are HTML elements on the page? That is, text, images, or other unchecked content, as long as they are related to HTML tags. (Verify/assertElementPresent)
- Need to check both the element and its text content? (Verify/assertText)
- There is no correct answer. It depends on your testing requirements. If in doubt, use assertText, because this is the most stringent type checkpoint. You can change it later, but at least you will not miss any potential failures.
- Verify/assertText is the most special type of test. Inconsistent HTML elements (tags) or text will cause the test to fail.
- Maybe your web designer often changes the page, and you don't want your test to fail when they change the page, because this is the expected periodic change.
- However, if you still need to check things on the page, such as paragraphs, title text, or images. In this case, you can use verify/assertElementPresent. This will ensure that a certain type of element exists (if you use XPath, you can ensure its existence relative to other objects on the page).
- But you don't care what the content is, you only care about a specific element, for example, a picture is in a specific position.
- With the passage of time and the accumulation of experience, how to decide to use is still very simple.
- Using the ID or name locator of an element is the most effective way in terms of test execution. It also makes your test code more readable, if the ID or name attributes in the page source code are friendly named.
- XPath statements take longer to process because the browser must run its XPath processor. In Internet Explorer 7, XPath is notoriously slow.
- It is very convenient to use the linked text for positioning, and it works well. This technique only applies to links.
- In addition, if the link text is likely to change frequently, using the <a> tag to locate the element will be a better choice.

- However, sometimes you must use XPath positioning. If a page element does not have an ID or name attribute, there is no choice but XPath positioning. (DOM locators are no longer commonly used because XPath can do better. DOM locators simply exist for legacy testing).
- Compared with ID or name attribute positioning, using XPath for positioning has a unique advantage. Using XPath (DOM), you can find an object relative to other objects on the page.
- For example, if there is a link that must exist in the second paragraph of the tag, you can use XPath to locate it. Using ID and name attribute positioning, you can only conclude that they exist on the specified page, without knowing the specific page location.
- If you have to test the image showing the company logo at the top of the page, XPath positioning may be a better choice.

☞ **Position dynamic elements**

- As mentioned earlier in the Test Type section, the page IDs of dynamic elements is listed differently in different page instances. E.g,

```
<a class="button" id="adminHomeForm" onclick="return SubmitForm('adminHomeForm',
'adminHomeForm:_ID38');" href="#">View Archived Allocation Events</a>
```

- This HTML anchor tag defines a button with an ID attribute of "adminHomeForm".
- Compared to most HTML tags, this is a fairly complex anchor tag, but it is still a static tag. Each time the page is loaded by the browser, the HTML will remain unchanged.
- Its ID remains the same in all page instances, that is, this UI element always has the same identifier when the page is displayed. Therefore, the test script (Selenium Server) that clicks this button is as follows:

```
selenium.click("adminHomeForm");
```

- However, your application may generate dynamic HTML identifiers. In different web page instances, the identifier changes.
- For example, the HTML element of a dynamic page might look like this:

```
<input id="addForm:_ID74:_ID75:0:_ID79:0:checkBox"
type="checkbox" name="addForm:_ID74:_ID75:0:_ID79:0:checkBox"
value="true"/>
```

- This is a check box, both id and name attributes are addForm: _ID74: _ID75: 0: _ID79: 0: checkBox. In this case, using standard positioning, the test script should look like this:

```
selenium.click("addForm:_ID74:_ID75:0:_ID79:0:checkBox");
```

- For dynamically generated identifiers, this approach does not work.
- The next time the page loads, the identifier will be a different value, and you will encounter an "element not found" error when executing the above script.
- To correct this problem, a simple solution is to use XPath positioning instead of ID locator. Therefore, for this check box, you can simply use

```
selenium.click("//input");
```

- Or, if it is not the first text input field on the page, try a more detailed XPath statement.

```
selenium.click("//input[3]");
or
selenium.click("//div/p[2]/input[3]");
```

- However, if you really need to use the ID to locate the element, you can change to a different solution. You can capture this ID of the website before using it, for example:

```
String[] checkboxids  = selenium.getAllFields();//Collect all input IDs on page.
for(String checkboxid:checkboxids) {
if(checkboxid.contains("addForm")) {
selenium.click(expectedText);
        }
        }
```

- This method works if the ID text of only one check box on the page is "expectedText".

☞ Positioning Ajax elements

- The best way to locate and verify AJAX elements is to use the Selenium 2.0 webdriver API, which specifically addresses some of the limitations of Selenium 1.0 testing AJAX elements.
- In Selenim 2.0, you can use the waitfor() method to wait for a page element to become available.
- This parameter is a By object used by WebDriver to achieve positioning. This is explained in detail in the WebDriver chapter.
- In Selenium 1.0 (Selenium-RC), to do this requires more coding, but it is not difficult. Check the element first, if it exists, wait for a predefined period of time, and then check again.
- This is performed within the loop. If a predetermined timeout is exceeded and the element does not exist, the loop is terminated.
- Let us consider a link on the page that implements the AJAX effect (link = ajaxLink), which can be handled using a loop:

```
//Loop initialization.
for (int second = 0;; second++) {
   //If loop is reached 60 seconds then break the loop.
if (second &gt;= 60) break;
   //Search for element "link=ajaxLink" and if available then break loop.
try { if (selenium.isElementPresent("link=ajaxLink")) break; } catch (Exception e) {}

//Pause for 1 second.
Thread.sleep(1000);
}
```

- This is certainly not the only solution. Ajax is a common topic. On the user forum, look up the previous discussion and see how others have done it.

☞ Encapsulate Selenium calls

- As with any programming, you need to use tool functions to handle functions that are repeated in the test code.
- One way to avoid duplication is to encapsulate the commonly used Selenium method calls. For example, when testing, you often click on elements on the page and wait for the page to load.

```
selenium.click(elementLocator);
selenium.waitForPageToLoad(waitPeriod);
```

- In order not to repeat the above code, you can write a wrapper method to achieve these two functions.

```
/**
* Clicks and Waits for page to load.
*
* paramelementLocator
* paramwaitPeriod
*/
public void clickAndWait(String elementLocator, String waitPeriod) {
selenium.click(elementLocator);
selenium.waitForPageToLoad(waitPeriod);
}
```

"Safe operation" to determine the existence of an element

- Another common method of encapsulating Selenium is to check the presence of elements on the page before performing further operations. This is sometimes referred to as "safe operation".
- For example, the following method can be used to implement a safe operation that depends on the presence of the desired element.

```
/**
* Selenum-RC -- Clicks on element only if it is available on page.
*
* paramelementLocator
*/
public void safeClick(String elementLocator) {
if(selenium.isElementPresent(elementLocator)) {
selenium.click(elementLocator);
    } else {
        //Using the TestNG API for logging
Reporter.log("Element: " +elementLocator+ ", is not available on page - "
                +selenium.getLocation());
    }
}
```

- The above example uses the Selenium 1.0 API, Selenium 2.0 also supports secure operations.

```
/**
* Selenium-WebDriver -- Clicks on element only if it is available on page.
*
* paramelementLocator
*/
public void safeClick(String elementLocator) {
```

```
WebElementwebElement = getDriver().findElement(By.XXXX(elementLocator));
if(webElement != null) {
selenium.click(elementLocator);
    } else {
        //Using the TestNG API for logging
Reporter.log("Element: " +elementLocator+ ", is not available on page - "
                + getDriver().getUrl());
    }
}
```

- In the second example, the 'XXXX' method is a placeholder and can be replaced with an element positioning method.
- The use of a safe method depends on the test developer's decision. Therefore, if the test needs to be continued, even if you know that some elements on the page are not found, you can use the secure method and send a message with missing elements to the log file.
- This is basically equivalent to implementing verification with a reporting mechanism, rather than an assertion that terminates execution upon failure.
- However, if the element must appear on the page in order to be able to perform further operations (such as a login button on the homepage of a portal), then the security method technique should not be used.

5.9 AUTOMATION TOOLS: SOAPUI, ROBOTIC PROCESS AUTOMATION (RPA), TOSCA, APPIUM.

5.9.1 SoapUI

SoapUI is the world's leading open-source testing platform. It is the most widely used automation tool for testing web services and web APIs of SOAP and REST interfaces. It is a boon for testers to test functional and non-functional testing, such as automated testing, functional, load testing, regression, simulation and mocking without hindrance because its user interface is very simple to use. It supports various standard protocols such as HTTP, HTTPS, REST, AMF, JDBC, SOAP, etc., that exchange information in structured data such as XML, plain text or JSON, etc. with the help of network services or web APIs in a computer.

Why we use SoapUI ?

- It is an important tool to test the Web domain, and it is an open-source, cross-platform as well as language independent that supports Eclipse, NetBeans, and IDEA. It allows the testers to test functional, non-functional testing, performance testing, regression testing, compilation, and load testing on various Web services and APIs.
- It is an important tool to test the Web domain, and it is an open-source, cross-platform as well as language independent that supports Eclipse, NetBeans, and IDEA. It allows the testers to test functional, non-functional testing, performance testing, regression testing, compilation, and load testing on various Web services and APIs.

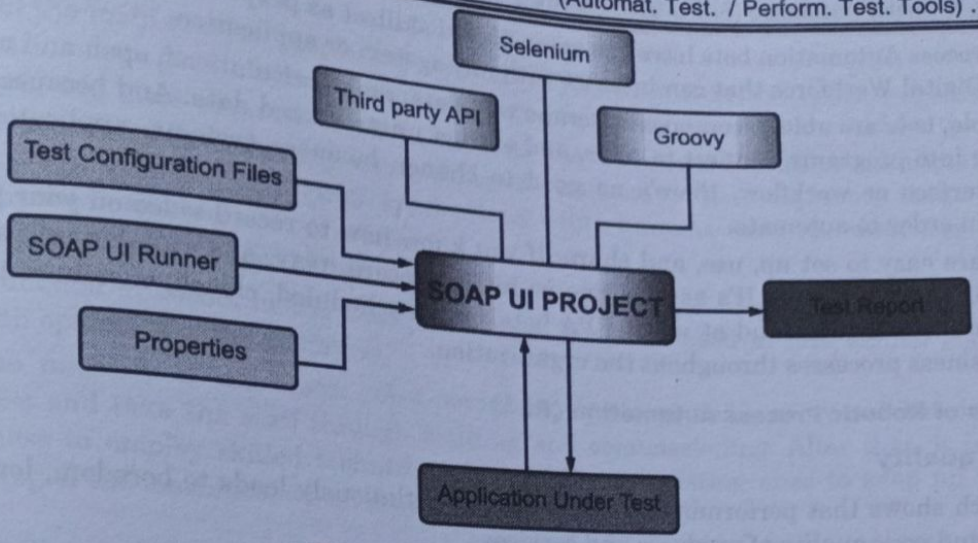


Fig. 5.9.1 : Soap-UI Architecture

☞ Advantages of SoapUI

(1) It provides a simple and user-friendly Graphical User Interface (GUI).
(2) Cross-platform desktop-based application.
(3) It supports all standard protocols and technologies such as HTTP, HTTPS, AMF, JDBC, SOAP, WSDL, etc.
(4) SoapUI costs less than all other test tools available in the market.
(5) It is also used as message broadcasting.
(6) It provides a fast and well-organized framework that generates lots of web services tests.
(7) It creates mocks where testers can test real applications.
(8) It supports drag and drops features to access script development.
(9) Transferring data from one response or source to different API calls without manual interaction in the SoapUI tool.
(10) It facilitates tester and developer teams to work together.
(11) SOAPUI tool provides the facility to get data from various sources of web service without developing any code.

☞ Disadvantages of SoapUI

(1) Security testing requires enhancements.
(2) The Mock response module should be more enhances and simplified.
(3) It takes longer to request big data and dual tasks to test web services.

5.9.2 Robotic Process Automation (RPA)

- Robotic Process Automation (RPA) is software technology that's easy for anyone to use to automate digital tasks. With RPA, software users create software robots, or "bots", that can learn, mimic, and then execute rules-based business processes.
- RPA automation enables users to create bots by observing human digital actions. Show your bots what to do, then let them do the work. Robotic Process Automation software bots can interact with any application or system the same way people do except that RPA bots can operate around the clock, nonstop, much faster and with 100% reliability and precision.

- Robotic Process Automation bots have the same digital skillset as people—and then some. Think of RPA bots as a Digital Workforce that can interact with any system or application.
- For example, bots are able to copy-paste, scrape web data, make calculations, open and move files, parse emails, log into programs, connect to APIs, and extract unstructured data. And because bots can adapt to any interface or workflow, there's no need to change business systems, applications, or existing processes in order to automate.
- RPA bots are easy to set up, use, and share. If you know how to record video on your phone, you'll be able to configure RPA bots. It's as intuitive as hitting record, play, and stop buttons and using drag-and-drop to move files around at work. RPA bots can be scheduled, cloned, customized, and shared to execute business processes throughout the organization.

☞ **Advantages of Robotic Process Automation (RPA)**

(1) Improved quality

- Research shows that performing the same tasks continuously leads to boredom, low concentration levels, and poor quality of products and services.
- Robots save employees from performing these repetitive tasks, by taking over production and accurately checking that all the products maintain high standards. Better quality products, in turn, create new and repeat opportunities for the business.

(2) Cuts down on costs

- Businesses lose a lot of income due to employees taking time off to attend to their issues, lunch breaks, sick leaves, and holidays among others.
- Robots, on the other hand, can work all year round without taking any breaks, which helps to raise production at much lower costs. They also help to save from employing other staff members to fill up positions left vacant by absent employees.

(3) Increase in sales

Freeing employees from performing repetitive tasks raises their morale making them more productive, with a better environment at the workplace and higher energy levels, there is more motivation to improve on their input, which helps to increase sales and have more satisfied clients.

(4) Eliminate risks

- Some industries have dangerous work environments that lead to injuries from chemical spills and others.
- Such industries also spend large amounts of money to ensure that the employees do not risk their health and lives while at work. By using robots to take over the dangerous and risky applications, you not only save your employees from risk, but you also save money from buying lots of security and safety equipment.

☞ **Disadvantages of Robotic Process Automation (RPA)**

1. High investment costs

- Robots do not come cheap, and the initial cost of deploying them may not be an easy thing to do. It is crucial to ensure that you have a comprehensive business plan before considering the deployment of automation at your workplace.
- You need to consider if deploying robots will increase your output and reduce expenses before deciding to put some income aside for capital expenditure.

2. **Job losses**
   - One of the main reasons people are resistant to robotic automation is the fear of job losses. On the contrary, robots do not take jobs away as there are always new roles created for the staff members.
   - Some of the most innovative robots, such as the Universal Robots cobots come with specialized programs that allow them to work alongside humans without replacing them. Robots, in most cases, take over the high risky applications while humans take on other, less risky tasks that benefit the business.
3. **Upskilling or sourcing for skilled staff**
   - Introduction of robots in the workplace comes with the need to hire skilled personnel to enable smooth operation of the machines.
   - In the majority of the cases, robot-manufacturing companies carry out the initial installation process and take the staff through training and commissioning. After that, it is either upon the business to employ skilled technicians or upskill the existing ones to keep up with the flawless working of the machines, which may attract extra costs.

5.9.3 Tosca

Tosca Automation Tool

- Being a test tool, Tosca has the ability to automate the functional and regression testing scenarios. It is also capable of mobile and API testing, which is now mandatory for any product delivery in AGILE mode.
- Tosca supports scripts less automation i.e., scripts and coding is not required to automate any scenario. So, anyone can learn the tool easily and start developing test cases. TOSCA supports its users to build efficient test cases in a methodologically way and provide detailed reports for management.

Tosca Key Features Are :

1. **Model-Based Testing Approach :** It's one of the significant features of Tosca as a test automation tool. It helps Tosca to gain leverage over other automation tools. Tosca creates a model of AUT (application under test) to create the test case without using of scripts.
2. **Risk-Based Testing Approach :** As per the name explains, this feature helps users to assess the risk with respect to test cases and allows them to identify the right set of test scripts to minimize the risks. Following different black box testing approaches such as boundary testing, equivalence partitioning, decision box, linear expansion, etc. are utilized to reduce the test script count by ensuring the functional risk coverage. After completion of test execution, risks are measured based on the test results and risk coverage.
3. **Script less test cases :** Tosca allows script less automation. It means test cases are created based on the modules which are added by drag and drop method, test data parameters, etc. after carefully incorporating the checkpoints. So, anybody can develop the test suite with minimum programming knowledge.
4. **Dynamic test data :** Test data can be stored separately in a central repository.
   Easy to the maintenance of test cases: In case of a change in application or data, it can be easily incorporated in the test suite by updating the centrally stored modules, library, and data. Distribute Execution : Tosca also provides a great feature to schedule and execute the test cases in different distributed systems in an unattended mode. It reduces the human efforts for testing.

Advantages of Tosca

1. Multiple Features in One Tool
2. No Scripting Required
3. Testing Methodology
4. Supports Multiple Platforms
5. Quality Vendor Support

Disadvantages of Tosca

1. It is highly expensive when compared with other automation tools
2. It is a heavy tool to maintain.
3. Provides less performance while scanning the application

5.9.4 Appium

- Appium is an open-source automation mobile testing tool, which is used to test the application. It is developed and supported by Sauce Labs to automate native and hybrid mobile apps. It is a cross-platform mobile automation tool, which means that it allows the same test to be run on multiple platforms.
- Multiple devices can be easily tested by Appium in parallel. In today's development area, the demand for mobile applications is high.
- Currently, people are converting their websites into mobile apps. Therefore, it is very important to know about mobile software automation testing technology and also stay connected with new technology. Appium is a mobile application testing tool that is currently trending in Mobile Automation Testing Technology.

Features of Appium

- Appium does not require application source code or library.
- Appium provides a strong and active community.
- Appium has multi-platform support i.e., it can run the same test cases on multiple platforms.
- Appium allows the parallel execution of test scripts.
- In Appium, a small change does not require re-installation of the application.
- Appium supports various languages like C#, Python, Java, Ruby, PHP, JavaScript with node.js, and many others that have Selenium client library.

Advantages of Appium

1. Appium is an open-source tool, which means it is freely available. It is easy to install.
2. It allows the automated testing of hybrid, native, and web applications.
3. Unlike other testing tools, you do not need to include any additional agents in your app to make Appium compatible with automation. It tests the same app, which is going to upload in App Store.
4. An additional feature added to Appium. Now it would support desktop application testing for windows as well along with mobile application testing.
5. Appium is a cross-platform, freely available mobile testing tool, which allows us the cross-platform mobile testing. This means you can test on multiple platforms (single API for both Android and IOS platforms).

Disadvantages of Appium

1. Along with some features and advantages, Appium has some drawbacks too, which are as follows-
2. Lack of detailed reports.
3. Since the tests depend on the remote web driver, so it is a bit slow.
4. It is not a limitation, but an overhead that Appium uses UI-Automator for Android that only supports Android SDK, API 16, or higher.

...Chapter Ends

❑❑❑

UNIT VI

CHAPTER 6

Testing Framework

University Prescribed Syllabus for the Academic Year 2022-2023

Testing Framework : Software Quality, Software Quality Dilemma, Achieving Software Quality, Software Quality Assurance. Elements of SQA, SQA Tasks, Goals and Metrics, Formal Approaches to SQA, Statistical Software Quality Assurance, Six Sigma for Software Engineering, ISO 9000 Quality Standards, SQA Plan, Total Quality Management, Product Quality Metrics, In process Quality Metrics, Software maintenance, Ishikawa's 7 basic tools, Flow Chart, Checklists, Pareto diagrams, Histogram, Run Charts, Scatter diagrams, Control chart, Cause Effect diagram. Defect Removal Effectiveness and Process.

6.1 INTRODUCTION TO SOFTWARE QUALITY

UQ. What Is Software Quality ?

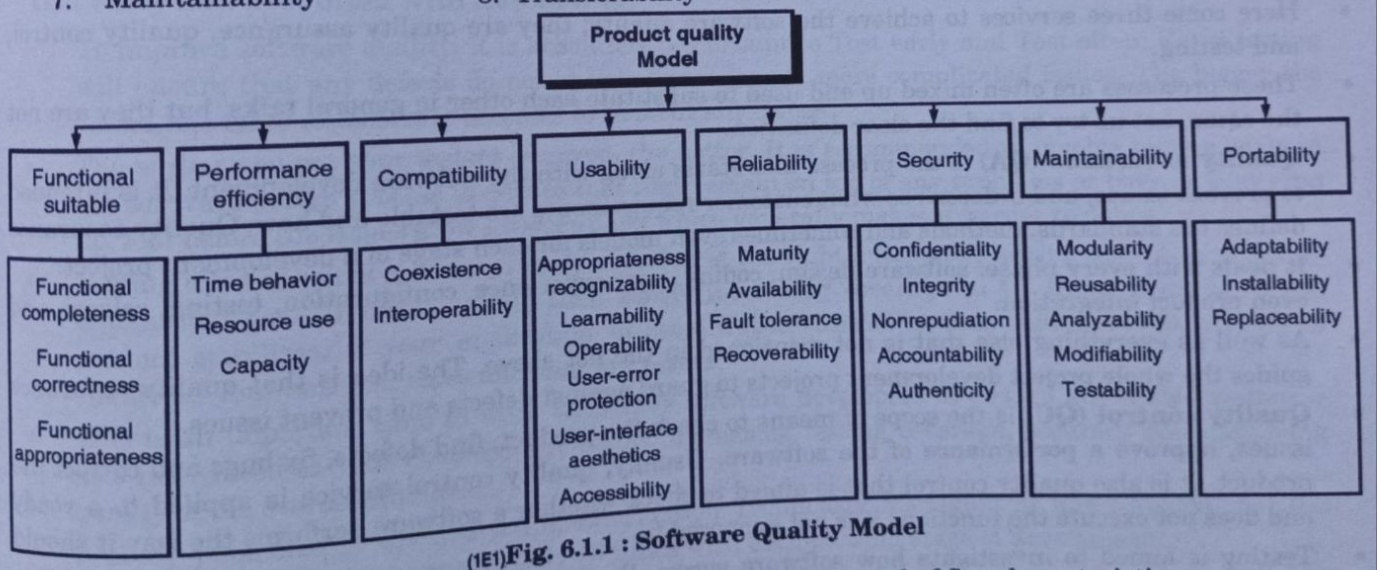
GQ. Explain Software Quality Management ? SPPU - Aug.19(Insem), May 18 (Endsem)

- **Software quality** is defined as a field of study and practice that describes the desirable attributes of software products. There are two main approaches to software quality: defect management and quality attributes.
- A software defect can be regarded as any failure to address end-user requirements. Common defects include missed or misunderstood requirements and errors in design, functional logic, data relationships, process timing, validity checking, and coding errors.
- The software defect management approach is based on counting and managing defects. Defects are commonly categorized by severity, and the numbers in each category are used for planning. More mature software development organizations use tools, such as defect leakage matrices (for counting the numbers of defects that pass through development phases prior to detection) and control charts, to measure and improve development process capability.

Software Quality characteristics

- This approach to software quality is best exemplified by fixed quality models, such as ISO/IEC 25010:2011. This standard describes a hierarchy of eight quality characteristics, each composed of sub-characteristics :

1. Functional suitability
2. Reliability
3. Operability
4. Performance efficiency
5. Security
6. Compatibility
7. Maintainability
8. Transferability



(1E1) **Fig. 6.1.1 : Software Quality Model**

- Additionally, the standard defines a quality-in-use model composed of five characteristics:
  1. Effectiveness
  2. Efficiency
  3. Satisfaction
  4. Safety
  5. Usability
- A fixed software quality model is often helpful for considering an overall understanding of software quality. In practice, the relative importance of particular software characteristics typically depends on software domain, product type, and intended usage. Thus, software characteristics should be defined for, and used to guide the development of, each product.

- Quality function deployment provides a process for developing products based on characteristics derived from user needs.

6.1.1 Brief History of Software Quality Dilemma

UQ. Brief History of Software Quality Dilemma ? SPPU - Aug.18 (Insem)
UQ. What is Software Quality Dilemma ? SPPU - Nov./Dec. 19 (Endsem)

- Software quality dilemma is a situation where there is confusion regarding what should we prioritize: a good quality work or a fast paced work.
- In software development, many a times there will be deadlines to achieve, in such cases software quality dilemma is bound to occur. A developer would have to choose between writing and optimized and well commented code or just get the job done without proper optimized or reviews.
- In same lines, many companies have to decide between regular reviews and expert opinions of a product for good software quality or bypass them to meet budgets and deadlines.
- Quality is a complex term comprising a lot of meanings in a single word. Saying the quality, we mean the product lacks defects, is vulnerable-free, it meets our requirements.
- The good is made durable and reliable enough to serve or perform the whole lifecycle smoothly. Yet, it is not the end, the quality also ensures that the same product is effective and efficient. Quite an exciting scope of senses, indeed.
- As you may guess, in this post, we are going to write about quality, not in its general sense, but about quality assurance in software development area.
- Software quality management is intended to ensure the business value of delivered software, meaning an application conforms to requirements for its reliability, efficiency, security, maintainability, and size. Moreover, all the parameters correlate with risk and cost management.
- Here come three services to achieve the software quality; they are quality assurance, quality control, and testing.
- These processes are often mixed up and used to substitute each other in general talks, but they are not the same. Let us try to find the clear definition.
- **Quality assurance (QA)** is the process that starts along with the software development. It is intended to prevent issues, and it describes what customer requirements are testable and how. Quality assurance defines the standards, methods and sometimes even models for each stage of a development project.
- It deals with every phase: software design, coding, user experience, configuration, testing, release and even product integration.
- As well as everything else that is not mentioned in the list above. The idea is that quality assurance guides the whole project development projects to guard against defects and prevent issues.
- **Quality control (QC)** is the scope of means to check the product, find defects, fix bugs and correct the issues, improve a performance of the software. Usually, quality control service is applied to a ready product. It is also quality control that is aimed to check whether a software performs the way it should and does not execute the functions it is not supposed to.
- Testing is aimed to investigate how software works. Engineers set up a testing plan, perform tests, analyse their results, and confirm tests' validity and their verification. Of course, experts create a lot of reports on the work they have done. The test can be automated, or manual; analysis - quantitative and qualitative, both software structure and its functions can be inspected to get the product that meets customer's requirements and users' expectations.
- To some extent, quality assurance overlaps with quality control and testing, but they should do. They are all software quality management services, which are not about breaking a software or bug finding.

- The main goal of QA is to prevent errors and avoid outage and to save your budget, efforts, and resources at every moment of your software lifecycle.
- Protect your products, don't waste your money on unnecessary software maintenance or support, include quality assurance in the software development process.

6.2 ACHIEVING GOOD SOFTWARE QUALITY

UQ. What parameters required for achieving good Software Quality ? **SPPU - Nov. 19 (Endsem)**

- How you manage software quality has become a vital element of every stage of project management. Are you continually on the lookout for ways to improve your software quality that is not going to break the bank ? Excellent software quality will enable cost effectiveness and superior performance to deliver your projects.
- Finding ways to implement effective testing strategies at the earliest possible stage will help you detect and solve defects. Solving problems at the earliest stage of project management creates a win-win scenario. Increased efficiency results in better quality software and reduced costs. Conversely, poor software quality exacerbates problems and can become a time-consuming and expensive exercise.
- Instead of spending lengthy periods of time fire-fighting software issues, you can concentrate on delivering a quality project. To help you increase efficiency and excellence for your next project we will explain 11 effective methods to improve software quality. These methods are aimed to provide you with assistance so you can deliver your next project with peace of mind in how well your software will operate.
- This article will help you and your project team take a complete assured approach to software development.

1. **Test early and Test often with Automation**
   - To improve software quality, it is absolutely paramount to Test early and Test often. Early testing will ensure that any defects do not snowball into larger, more complicated issues. The bigger the defect, the more expensive it becomes to iron out any issues.
   - The earlier you get your testers involved, the better. It is recommended to involve testers early in the software design process to ensure that they remain on top of any problems or bugs as they crop up, and before the issues grow exponentially which generally makes it harder to debug.
   - Testing often requires a focus on early adoption of the right automated testing discipline. Start by automating non UI tests initially then slowly increasing coverage to UI based tests when the product stabilises. If your application utilises Webservices/APIs then automate these tests to ensure all your business rules and logic are tested.
   - This is an important time to work with your software developers to ensure automated testing is also introduced to your development teams, increasing testing coverage, accuracy and improving quality of the overall product.
   - A study published in the Journal of Information Technology Management has revealed that the cost to rectify a bug increases roughly 10 times with each passing stage of development.
   - **For Example :** An error that costs $100 to rectify in the business requirements stage would cost $1000 to rectify in the system requirements stage, $10,000 in the high-level design stage and $100,000 in the implementation stage.
2. **Implement quality controls from the beginning**
   - Testers can monitor quality controls and create awareness in partnership with developers to ensure standards are continually being met. Quality control starts from the beginning, which is an ongoing process throughout the delivery.

- A good relationship between testers and developers can help the project software strategy develop effectively. A systematic methodology in quality control can ensure that coding errors and bugs are dealt with effectively, following a structured process.

3. Echo the importance of quality assurance through the entire software development process

- We have identified how important testing is at the beginning of software development; however, the testing does not stop there. Quality assurance should be ever-present throughout the software development process.
- Quality Assurance is governance provided by the project team that instils confidence in the overall software quality. Assurance testing oversees and validates the processes used in order to deliver outcomes have been tracked and are functioning. Testing should be repeated as each development element is applied. Think of it as layering a cake. After every layer is added, the cake should be tasted and tested again.

4. Encourage innovations

- It is important that testing structures and quality measures are in place, however, there should always be room for innovation. A great way to allow for innovation is to automate testing where possible to minimise time spent on controls.
- Innovations are so important because they can lead to improvements in software quality that have the capability to transform how projects are delivered. Research and development (R&D) should be encouraged. Empower teams to explore, experiment and investigate continuously. Also, ensure that advancements in innovation are duly rewarded. They have the capacity to transcend your software quality and deliver projects with a competitive advantage over the competition.

5. Communication is key

- For any relationship to be successful whether it's personal or business, communication is key. To improve software quality it is important that all parties to the project have full information through fluid communication channels.
- Fluid communication can take many forms. It can be as simple as having clear, consistent KPIs that show how software quality is measured at every step of the development process. It is important that all team members, regardless of seniority have access to KPIs to keep the entire team on the same page. Another important aspect of fluid communication is that all parties have the opportunity to provide feedback to the team to ensure that all expectations are being met.
- It is also important to keep all stakeholders in the loop and not isolate team members from the vendors or end user of the software. Isolation can cause rifts and can often mean that the project is delayed or may not deliver on the goals expected by senior management.

6. Plan for a changeable environment

- Software contains so many variables and is in continuous evolution. It relies on several different external factors such as web browsers, hardware, libraries, and operating systems.
- These constant external factors mean that software development must be consistently monitored using checks and balances to certify that it remains in stride with its immediate environment.
- It is important to acknowledge that software is interdependent on these external factors. Accepting this interdependence means that you can plan ahead. It allows you to have the software quality tested, at each step of the process, against external variables, to see how it holds up. The end result is that you will prevent software dissonance and maintain software quality.

7. Take the attitude of creating products not projects

- This step is a reflection of the attitude of your team. Creating a project indicates to your team that you are producing a finite outcome. However, we are well aware that software is changeable. If you produce a finite outcome, before long the software quality will not stand up against its environment.

- Instead, if your team takes the mind-set that they are creating a product it is more likely that they will deliver software quality that is adaptable to change and can stand the test of time. Focus on delivering continuous small progressions rather than one final end project and your team will deliver an increase in quality.

8. **Have a risk register**

- A risk register is a fantastic management tool to manage risks. A risk register is more synonymous with financial auditing; however it is still a vital element in software development.
- A risk register will provide everybody aligned on a software project a list of clearly identified risks and then assess them in regards to the importance of delivering the project. A risk register works well for software quality because its creation actively leads to risk mitigation.

A software development risk register must :

- Describe the risk.
- Recognise when the risk was identified.
- Acknowledge the chance of the risk occurring and its mitigation.
- Understand the severity of the impact of the risk.
- Identify who assesses and actions the risk.
- Relays the response if the risk materialises.
- Gives the status of each risk.
- Measures the negative impact of each risk.
- Prioritises the risks ranked on probability and gravity.

9. **Producing software quality requires long-term thinking and strategy**

Long-term thinking produces software quality because decisions are made to satisfy lasting issues. Here are the advantages of long-term thinking for producing software quality:

1. Doing it right first means you don't have to spend time doing it over.
2. Placing as much importance on architecture and design as coding ensures the veracity of your project.
3. Creating coding standards with long-term vision eliminates unnecessary mistakes.
4. Effective peer review ensures errors are minimised even though it may seem time-consuming at that particular moment.
5. Testing often allows you to plan further ahead with certainty that errors and bugs have been fixed.
6. Project leaders need to ensure that short-term gains and immediate gratification do not compromise long-term strategy. Effective planning will make sure all stakeholders prioritise software quality.

10. **Outline your deliverables**

- From the outset of your project it is imperative that your team outline what they are going to deliver. A clear and concise plan of what the project will deliver helps ensure there is an emphasis on quality from the outset.
- It also ensures that budgets, resources, and time are aligned correctly to deliver quality. Without clear deliverables it is likely shortcuts will be taken to meet budgets and deadlines. Ultimately, this will compromise the quality of the software delivered at the end of the project.

11. **Review, revise and remember**

Producing software quality is not a coincidence. This is why you must always do the following three things:

- **Review :** Testing often is a pillar of ensuring software quality. It ensures that standards are continuously met and bugs, errors and distractions can be fixed before they spiral out of control.

- **Revise :** Study what has worked throughout the software process. Utilise what is working and see if innovation can transcend your software quality even further.
- **Remember :** When you deliver quality remember what worked well and did not work well. Keep an updated record of both the positives and negatives of any given project and turn to it frequently when you start the next project from scratch.

6.3 SOFTWARE QUALITY ASSURANCE

UQ. How to maintain Software Quality Assurance ? **SPPU - Nov. 18 (End-sem)**

- Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards suitable for the project and implemented correctly.
- Software Quality Assurance is a process which works parallel to development of a software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of an Umbrella activity that is applied.
- There is no one universal definition of software quality. This is because of the complexity caused by the three or more participants affected by the quality of software, namely, customer, developer and stakeholders.
- The issue is whose views, expectations and aspirations are to be considered supreme. The majority hold that customer satisfaction should be the goal for measuring software quality.
- The customer may be satisfied though with software, the quality of which cannot be considered the best by other standards.
- The software quality definition is based on the following:
  1. Customer focus and customer satisfaction
  2. Functional and performance requirement
  3. Ease of learning, use and maintainability
  4. Adherence to development standards
- Customer satisfaction largely depends on meeting functional and performance requirements and ease of operations. Adhering to development standards ensures to a great extent the achievement of these goals.
- Software quality is defined as the quality that ensures customer satisfaction by offering all the customer deliverables on performance, standards and ease of operations. The definition is applicable for software as well as for a generic software product.

Software Quality Assurance (SQA)

- Software quality assurance is a planned effort to ensure that a software product fulfills these criteria and has additional attributes specific to the project, e.g., portability, efficiency, reusability, and flexibility.
- It is the collection of activities and functions used to monitor and control a software project so that specific objectives are achieved with the desired level of confidence.
- It is not the sole responsibility of the software quality assurance group but is determined by the consensus of the project manager, project leader, project personnel, and users.
- A formal definition of software quality assurance is that is 'the systematic activities providing evidence of the fitness for use of the total software product."
- Software quality assurance is achieved through the use of established guidelines throughout the software process.

Software Quality Assurance has :

1. A quality management approach
2. Formal technical reviews
3. Multi testing strategy
4. Effective software engineering technology
5. Measurement and reporting mechanism

6.3.1 Elements in SQA

UQ. What are different elements in SQA ? **SPPU - Nov. 18 (Endsem)**

1. Software engineering Standards
2. Technical reviews and audits
3. Software Testing for quality control
4. Error collection and analysis
5. Change management
6. Educational programs
7. Vendor management
8. Security management
9. Safety
10. Risk management

6.3.2 Task Goal and Metric in SQA

UQ. What are different task goal and metric in SQA ? **SPPU - May 19 (Endsem)**

Goals, Attributes and Metrics

- **Requirement quality :** The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow. SQA must ensure that the software team has properly reviewed the requirements model to achieve a high level of quality.
- **Design quality :** Every element of the design model should be assessed y the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.
- **Code quality :** Source code and related work products must conform to local coding standards and exhibit characteristics that will facilitate maintainability.
- **Quality control effectiveness :** A software team should apply limited resources in a way that has the highest likelihood of achieving a high–quality result. SQA analyzes the allocation of resources for reviews and testing to assess whether they are being allocated in the most effective manner.

Sr. No.	Goal	Attribute	Metric
1.	Requirement quality	Ambiguilty	Number of ambiginous modifiers (e.g. many, large, human-friendly)
		Completeness	Number of TBA, TBD
		Understandability	Number of sections/subsections
		Volatility	Number of changes per requirement Time (by activity) when change is requested.
		Traceability	Number of requirements not traceable to design/code.
		Model clarity	Number of UML models. Number of descriptive pages per model. Number of UML errors.

Sr. No.	Goal	Attribute	Metric
2.	Design qualtiy	Architecutral integrty	Existance of architectural model
		Component completeness	Number of components that trace to architectural model
			Complextiy of procedural design
		Interface complexity	Average number of pick to get to a typical function or content
		Patterns	Layot appropriateness
			Number of patterns used.
3.	Code qualtiy	Complexity	Cyclomatic complexity
		Maintainability	
			Design factors
		Understanability	Percent internal comments
			Variable naming conventions
		Reusability	Percent reused components
		Documentation	Readability index
4.	QC effectiveness	Resource allocation	Staff hour percentage per activity
		Completion rate	Actual vs. budgeted completion time
		Review effectiveness	See review metrics
		Testing effectiveness	Number of errors found and criticality
			Effort required to connect an error
			Origin of error

6.3.3 Formal Approaches to SQA

UQ. Formal Approaches to SQA, Statistical Software Quality Assurance?
SPPU - May 19 (Endsem)

Abbreviated as SQAP, the software quality assurance plan comprises of the procedures, techniques, and tools that are employed to make sure that a product or service aligns with the requirements defined in the SRS (software requirement specification).

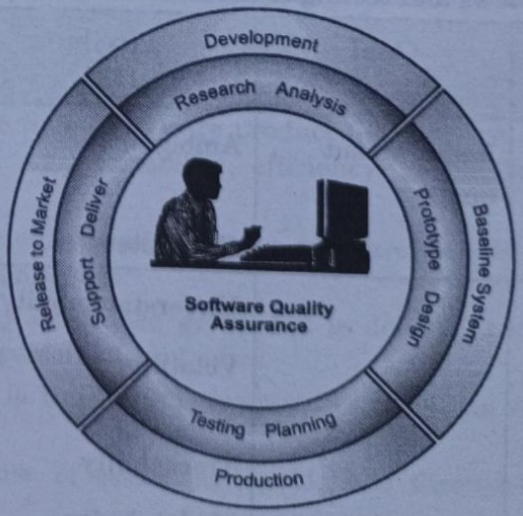


Fig. 6.3.1 : Software Quality Assurance Plan

- The plan identifies the SQA responsibilities of a team, lists the areas that need to be reviewed and audited. It also identifies the SQA work products.
- The SQA plan document consists of the below sections :
  - Purpose section
  - Reference section
  - Software configuration management section
  - Problem reporting and corrective action section
  - Tools, technologies and methodologies section
  - Code control section
  - Records : Collection, maintenance and retention section.

6.4 SIX SIGMA FOR SOFTWARE ENGINEERING

UQ. What is Six Sigma ? SPPU - May 18, Nov. 19 (Insem)

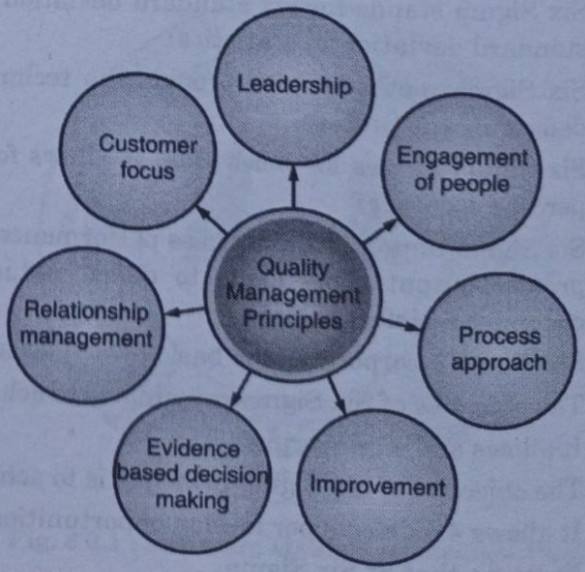
- Six Sigma is one of the most popular quality methods lately. It is the rating that signifies "best in class," with only 3.4 defects per million units or operations (DPMO). Its concept works and results in remarkable and tangible quality improvements when implemented wisely. Today, Six Sigma processes are being executed in a vast array of organization and in a wide variety of functions.
- Fuelled by its success at large companies such as Motorola, General electric, Sony, and Allied Signal, the methodology is proving to be much than just a quality initiative. Why are these large companies embracing Six Sigma? What makes this methodology different from the others?
- The goal of Six Sigma is not to achieve six sigma levels of quality, but to improve profitability. Prior to Six Sigma, improvements brought about by quality programs, such as Total Quality Management (TOM) and ISO 9000, usually had no visible impact on a company's net income. In general, the consequences of immeasurable improvement and invisible impact caused these quality programs gradually to be.
- Six Sigma was originally developed as a set of practices designed to improve manufacturing processes and eliminate defects, but its application was subsequently extended to other types of business processes as well In Six Sigma, a defect is defined as anything that lead to customer dissatisfaction.
- Six Sigma stands for six standard deviation from mean (sigma is three Greek letter used to represent standard deviation in statistics).
- Six Sigma methodologies provide the techniques and tools to improve the capability and reduce the defects in any process.
- Six Sigma strives for perfection. It allows for only 3.4 defects per million Opportunities (or 99.999666 percent accuracy)
- Six Sigma improves the process performance decrease variation and maintains consistent quality of the process output. This leads to defect reduction and improvements in profits, product quality and customer satisfaction.
- Six Sigma incorporates the basic principles and techniques used in business, statistics and engineering.
- The objective of Six Sigma principle is to achieve zero defects products/ process.
- It allows and engineering.
- The objective of Six Sigma principle is to achieve zero defects products/ process.
- It allows 4.4 defects per million opportunities.
- Features that St Six Sigma

- Apart from previous quality improvement initiatives include :
  - A clear focus on achieving measurable and quantifiable financial returns from any Six Sigma project.
  - An increased emphasis on strong and passionate management leadership and support.
  - A special infrastructure of "Champions, "Master black Belts, "black Belts", etc. to lead and implement the Six Sigma approach.
  - A clear commitment to making decisions on the basis of verifiable data, rather than assumptions and guesswork.
- Sigma Levels
  - 1 sigma = 690, 000 DPMO = 31% efficiency
  - 2 sigma = 308,000 DPMO= 69.2% efficiency
  - 3 sigma = 66,800 DPMO= 93.32% efficiency
  - 4 sigma = 6,210 DPMO = 99.379 % efficiency
  - 5 sigma = 230 DPMO = 99.977 % efficiency
  - 6 sigma = 3.4 DPMO = 99.9997 % efficiency

6.5 ISO 9000 STANDARDS

UQ. What are different ISO 9000 Standards ? **SPPU - May 18 (Insem)**

- ISO 9001: 2008 Quality management systems : Requirements is intended for use in any organization regardless of size, type or product (including service). It provides a number of requirements which an organization needs to fulfill to achieve customer satisfaction through consistent products and services which meet customer expectations. It includes a requirement for continual (i.e., planned) improvement of the Quality Management System, for which ISO 9004:2004 provides many hints.
- This is the only implementation for which third party auditors can grant certification. It should be noted that certification is not described as any of the 'needs' of an organization as a driver for using ISO 9001 but does recognize that it may be used for such a purpose.
- ISO 9004:2000 Quality management systems - Guidelines for performance improvements covers continual improvement. This gives you advice on what you could do to enhance a mature system. This document very specifically states that it is not intended as a guide to implementation.
- There are many more standards in the ISO 9001 series, many of them not even carrying "ISO 9000" numbers. For example, some standards in the 10,000 range are considered part of the 9000 group: ISO 10007: 1995 discusses configuration management, which for most organizations is just one element of a complete management system. The emphasis on certification tends to overshadow the fact that there is an entire family of ISO 9000 standards..
- Organizations stand to obtain the greatest value when the standards in the new core series are used in an integrated manner, both with each other and with the other standards making up the ISO 9000 family as a whole.



(1E3)**Fig. 6.5.1 : ISO Standard Principles**

- Note that the previous members of the ISO 9000 series 9002 and 9003 have been integrated into 9001. In most cases, an organization claiming to be "ISO 9000 registered" is referring to ISO 9001.

6.6 SOFTWARE QUALITY ASSURANCE PLAN

UQ. Explain in detail Software Quality Assurance Plan ? **SPPU - May 18 (Insem)**

- The software quality assurance (SQA) plan is an outline of quality measures to ensure quality levels within a software development effort. The plan is used as a baseline to compare the levels of quality during development with the planned levels of quality. If the levels of quality are not within the planned quality levels management will respond appropriately as documented within the plan.
- The plan provides the framework and guidelines for development of understandable and maintainable code. These Ingredients help ensure the quality sought in a software project, A SQA plan also provides the procedures for ensuring that quality software will be produced or maintained in house or under contract.
- These procedures affect planning. Designing, writing, testing, documenting, strong, and maintaining computer software. It should be organized in this way because the plan ensures the quality of the software rather than describing specific procedures for developing and maintaining the software.
- In the management approval process, management relinquishes tight control over software quality to the SQA plan administrator in exchange for improved software quality. Software quality is often left to software developers. Quality is desirable.
- But management may express concern as to the cost of a formal SQA plan. Staff should be aware that management views the program as a means of ensuring software quality, and not as an end in itself.
- To address management concerns, software life cycle costs should be formally estimated for projects implemented both with and without a formal SQA plan. In general, implementing a formal SQA plan makes economic and management sense.
- The SQA Plan helps to lay down the steps towards the quality goals of the organization. A standard for SQA plan gives details and templates on all activities, which have become part of the standard and which ensure quality standards implementation.
- Testing activity needs Test Plan likewise SQA activity also needs a plan which is called SQA plan.
- The goal of SQA plan is to craft planning processes and procedures to ensure products manufactured, or the service delivered by the organization are of exceptional quality.
- During project planning, Test Manager makes an SQA plan where SQA audit is scheduled periodically.
- The documentation of SQA plan includes
  - Project plan
  - Models of data, classes and objects, processes, design, architecture
  - Software Requirement Specifications (SRS)
  - Test plans for testing SRS
  - Users help documentation, manuals, online help, etc.
  - Reviews and audits
- The SQA process is made up of several activities. Some are organization specific, some are software specific and some are customer specific. It helps effective application of methods, tools, test plans and standards towards the goal of software quality. It also include measures, measurement and metric for building a quality in the organization.

6.7 INTRODUCTION TO TOTAL QUALITY MANAGEMENT

UQ. What Is Total Quality Management ? **SPPU - Oct. 19 (Insem)**

- Total quality management (TQM) is the continual process of detecting and reducing or eliminating errors in manufacturing, streamlining supply chain management, improving the customer experience, and ensuring that employees are up to speed with training.
- Total quality management aims to hold all parties involved in the production process accountable for the overall quality of the final product or service.
- TQM was developed by William Deming, a management consultant whose work had a great impact on Japanese manufacturing.
- While TQM shares much in common with the Six Sigma improvement process, it is not the same as Six Sigma. TQM focuses on ensuring that internal guidelines and process standards reduce errors, while Six Sigma looks to reduce defects.
- Total quality management (TQM) is a structured approach to overall organizational management. The focus of the process is to improve the quality of an organization's outputs, including goods and services, through continual improvement of internal practices.
- The standards set as part of the TQM approach can reflect both internal priorities and any industry standards currently in place.
- Industry standards can be defined at multiple levels and may include adherence to various laws and regulations governing the operation of the particular business.
- Industry standards can also include the production of items to an understood norm, even if the norm is not backed by official regulations.

6.8 PRODUCT QUALITY METRICS

UQ. Write in brief Product Quality Metrics ? **SPPU - Aug 17 (Insem), May 18 (Endsem)**

- Software metrics can be classified into three categories :
  1. **Product metrics :** Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.
  2. **Process metrics :** These characteristics can be used to improve the development and maintenance activities of the software.
  3. **Project metrics :** This metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.
- Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project metrics.
- Software quality metrics are a subset of software metrics that focus on the quality aspects of the product, process, and project. These are more closely associated with process and product metrics than with project metrics.
- Software quality metrics can be further divided into three categories :
  1. Product quality metrics
  2. In-process quality metrics
  3. Maintenance quality metrics

This metrics include the following :

(a) Mean Time to Failure (b) Defect Density
(c) Customer Problems (d) Customer Satisfaction

(a) Mean Time to Failure : It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.

(b) Defect Density : It measures the defects relative to the software size expressed as lines of code or function point, etc. i.e., it measures code quality per unit. This metric is used in many commercial software systems.

(c) Customer Problems : It measures the problems that customers encounter when using the product. It contains the customer's perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.

- The problems metric is usually expressed in terms of Problems per User-Month (PUM).
- PUM is usually calculated for each month after the software is released to the market, and also for monthly averages by year.

(d) Customer Satisfaction : Customer satisfaction is often measured by customer survey data through the five-point scale :

(i) Very satisfied (ii) Satisfied (iii) Neutral
(iv) Dissatisfied (v) Very dissatisfied

- Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys.
- Based on the five-point-scale data, several metrics with slight variations can be constructed and used, depending on the purpose of analysis. For example :
  1. Percent of completely satisfied customers
  2. Percent of satisfied customers
  3. Percent of dis-satisfied customers
  4. Percent of non-satisfied customers

  Usually, this percent satisfaction is used.

6.9 IN PROCESS QUALITY METRICS

UQ. What is in process Quality Metrics ? SPPU - Nov./Dec. 19 (Endsem)

In process quality metrics deals with the tracking of defect arrival during formal machine testing for some organizations. This metric includes :

1. Defect density during machine testing
2. Defect arrival pattern during machine testing
3. Phase-based defect removal pattern
4. Defect removal effectiveness

1. Defect density during machine testing

- Defect rate during formal machine testing (testing after code is integrated into the system library) is correlated with the defect rate in the field.
- Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process, unless the higher testing defect rate is due to an extraordinary testing effort.

- This simple metric of defects per KLOC or function point is a good indicator of quality, while the software is still being tested. It is especially useful to monitor subsequent releases of a product in the same development organization.

2. Defect arrival pattern during machine testing

The overall defect density during testing will provide only the summary of the defects. The pattern of defect arrivals gives more information about different quality levels in the field. It includes the following :

(i) The defect arrivals or defects reported during the testing phase by time interval (e.g., week). Here all of which will not be valid defects.

(ii) The pattern of valid defect arrivals when problem determination is done on the reported problems. This is the true defect pattern.

(iii) The pattern of defect backlog overtime. This metric is needed because development organizations cannot investigate and fix all the reported problems immediately. This is a workload statement as well as a quality statement. If the defect backlog is large at the end of the development cycle and a lot of fixes have yet to be integrated into the system, the stability of the system (hence its quality) will be affected. Retesting (regression test) is needed to ensure that targeted product quality levels are reached.

3. Phase-based defect removal pattern

- This is an extension of the defect density metric during testing. In addition to testing, it tracks the defects at all phases of the development cycle, including the design reviews, code inspections, and formal verifications before testing.
- Because a large percentage of programming defects is related to design problems, conducting formal reviews, or functional verifications to enhance the defect removal capability of the process at the front-end reduces error in the software.
- The pattern of phase-based defect removal reflects the overall defect removal ability of the development process.
- With regard to the metrics for the design and coding phases, in addition to defect rates, many development organizations use metrics such as inspection coverage and inspection effort for in-process quality management.

4. Defect removal effectiveness

- It can be defined as follows:

$$\text{DRE} = \frac{\text{Defect removed during a development phase}}{\text{Defects latent in the product}} \times 100\ \%$$

- This metric can be calculated for the entire development process, for the front-end before code integration and for each phase.
- It is called early defect removal when used for the front-end and phase effectiveness for specific phases.
- The higher the value of the metric, the more effective the development process and the fewer the defects passed to the next phase or to the field. This metric is a key concept of the defect removal model for software development.

6.10 SOFTWARE MAINTENANCE

UQ. What is Software Quality Maintenance Metrics? **SPPU - Nov./Dec.18(Insem)**

- Although much cannot be done to alter the quality of the product during this phase, following are the fixes that can be carried out to eliminate the defects as soon as possible with excellent fix quality.

1. Fix backlog and backlog management index

2. Fix response time and fix responsiveness
3. Percent delinquent fixes

☞ **Percent Delinquent Fixes**

$$\frac{\text{Number of fixes that exceeded the response time criteria by ceverity level}}{\text{Number of fixes delivered in a specified time}} \times 100\ \%\ \text{Fix quality}$$

(1) Fix backlog and backlog management index

- Fix backlog is related to the rate of defect arrivals and the rate at which fixes for reported problems become available. It is a simple count of reported problems that remain at the end of each month or each week.
- Using it in the format of a trend chart, this metric can provide meaningful information for managing the maintenance process.
- Backlog Management Index (BMI) is used to manage the backlog of open and unresolved problems.

$$\text{BMI} = \frac{\text{Number of problems closed during the month}}{\text{Number of problems arrived during the month}} \times 100\ \%$$

- If BMI is larger than 100, it means the backlog is reduced. If BMI is less than 100, then the backlog increased.

(2) Fix response time and fix responsiveness

- The fix response time metric is usually calculated as the mean time of all problems from open to close. Short fix response time leads to customer satisfaction.
- The important elements of fix responsiveness are customer expectations, the agreed-to fix time, and the ability to meet one's commitment to the customer.

(3) Percent delinquent fixes

It is calculated as follows :

$$\text{Percent Delinquent Fixes} = \frac{\text{Number of fixes that exceeded the response time criteria by veverity level}}{\text{Number of fixes delivered in a specified time}} \times 100\ \%$$

☞ **Fix Quality**

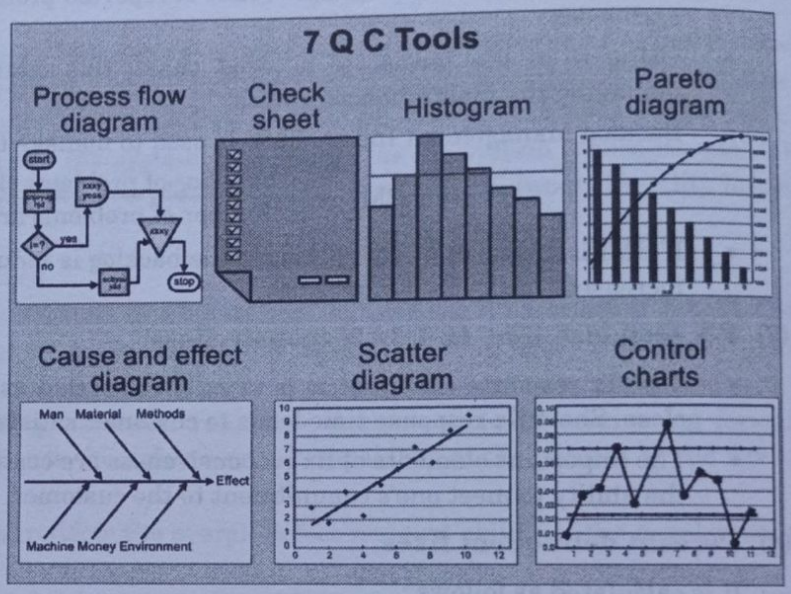
- Fix quality or the number of defective fixes is another important quality metric for the maintenance phase.
- A fix is defective if it did not fix the reported problem, or if it fixed the original problem but injected a new defect. For mission-critical software, defective fixes are detrimental to customer satisfaction.
- The metric of percent defective fixes is the percentage of all fixes in a time interval that is defective.
- A defective fix can be recorded in two ways: Record it in the month it was discovered or record it in the month the fix was delivered.
- The first is a customer measure; the second is a process measure. The difference between the two dates is the latent period of the defective fix.
- Usually the longer the latency, the more will be the customers that get affected. If the number of defects is large, then the small value of the percentage metric will show an optimistic picture.
- The quality goal for the maintenance process, of course, is zero defective fixes without delinquency.

6.11 ISHIKAWA'S 7 BASIC TOOLS

UQ. What are Ishikawa's 7 basic tools ? **SPPU - May 18 (Endsem)**

UQ. Explain the Terms Checklists, Pareto diagrams, Histogram, Run Charts, Scatter diagrams, Control chart, Cause Effect diagram. **SPPU - Aug. 17 (Insem), May 18 (Endsem), May 19 (Insem)**

- Dr. Kaoru Ishikawa was first total quality management guru, who has been associated with the development and advocacy of using the seven quality control (QC) tools in the organizations for problem solving and process improvements.
- Seven old quality control tools are a set of the QC tools that can be used for improving the performance of the production processes, from the first step of producing a product or service to the last stage of production. So, the general purpose of this paper was to introduce these 7 QC tools.
- This study found that these tools have the significant roles to monitor, obtain, analyze data for detecting and solving the problems of production processes, in order to facilitate the achievement of performance excellence in the organizations.
- There are seven basic quality tools, which can assist an organization for problem solving and process improvements.



(1F1)**Fig. 6.11.1 : Ishikawa 7 Basic Tool for Quality Control**

- The first guru who proposed seven basic tools was Dr. Kaoru Ishikawa in 1968, by publishing a book entitled "Gemba no QC Shuho" that was concerned managing quality through techniques and practices for Japanese firms.
- It was intended to be applied for "self-study, training of employees by foremen or in QC reading groups in Japan. It is in this book that the seven basic quality control tools were first proposed. valuable resource when applying the seven basic tools (Omachonu and Ross, 2004).
- These seven basic quality control tools, which introduced by Dr. Ishikawa, are :
  (1) Check sheets; (2) Graphs (Trend Analysis);
  (3) Histograms; (4) Pareto charts;
  (5) Cause-and-effect diagrams; (6) Scatter diagrams; (7) Control charts.
- Fig. 6.11.1 indicates the relationships among these seven tools and their utilizations for the identification and analysis of improvement of quality.

6.12 CHECK SHEET/CHECKLIST

- Check sheets are simple forms with certain formats that can aid the user to record data in an firm systematically.
- Data are "collected and tabulated" on the check sheet to record the frequency of specific events during a data collection period.

- They prepare a "consistent, effective, and economical approach" that can be applied in the auditing of quality assurance for reviewing and to follow the steps in a particular process. Also, they help the user to arrange the data for the utilization later.
- The main advantages of check sheets are to be very easily to apply and understand, and it can make a clear picture of the situation and condition of the organization.
- They are efficient and powerful tools to identify frequently problems, but they don't have effective ability to analyze the quality problem into the workplace. The chech sheets are in several, three major types are such as Defect-location check sheets; tally check sheets, and; defect-cause check sheets .

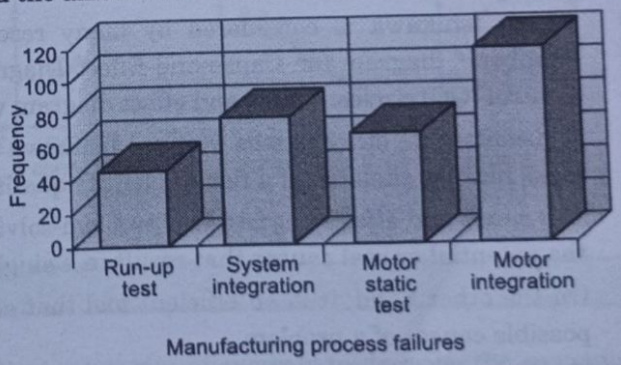
Telephone interruptions

Reason	Day					
	Mon	Tues	Wed	Thurs	Fri	Total
Wrong number	𝍸	II	I	𝍸	𝍸 II	20
Info request	II	II	II	II	I	10
boss	𝍸	II	𝍸 II	I	IIII	19
Total	12	6	10	8	13	49

(1F2)**Fig. 6.12.1 : Check sheet Tally for telephone interruptions**

6.13 HISTOGRAM

- Histogram is very useful tool to describe a sense of the frequency distribution of observed values of a variable. It is a type of bar chart that visualizes both attribute and variable data of a product or process, also assists users to show the distribution of data and the amount of variation within a process.
- It displays the different measures of central tendency (mean, mode, and average). It should be designed properly for those working into the operation process can easily utilize and understand them.
- Also, a histogram can be applied to investigate and identify the underlying distribution of the variable being explored. Fig. 6.13.1 illustrates a histogram of the frequency of defects in a manufacturing process.

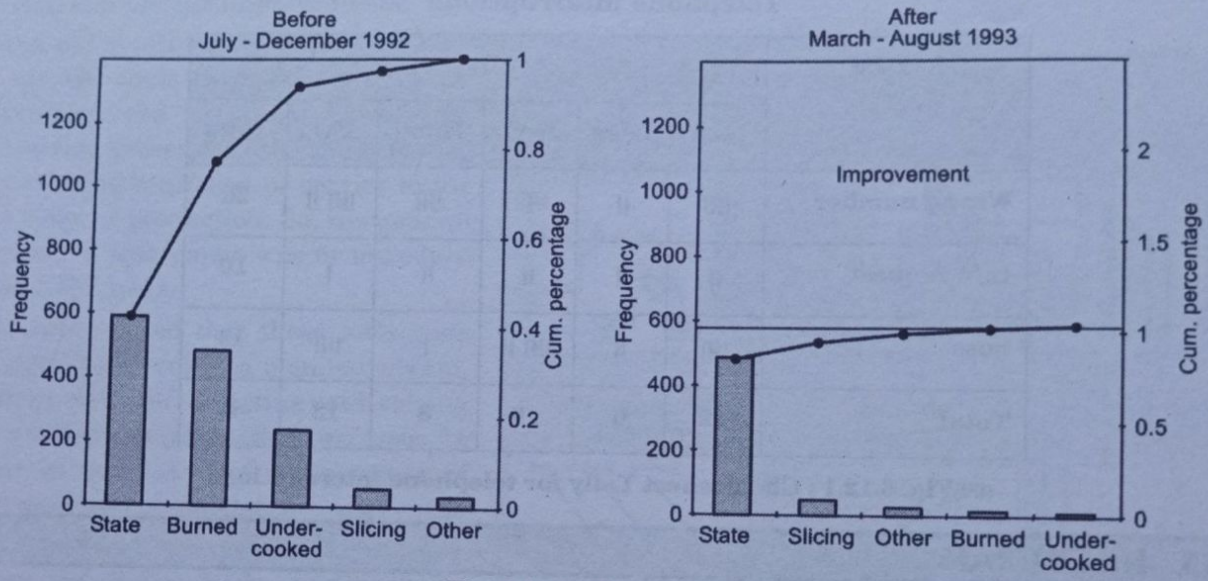


(1F3)**Fig. 6.13.1 : Histogram for variables**

6.14 PARETO ANALYSIS

- It introduced by an Italian economist, named Vilfredo Pareto, who worked with income and other unequal distributions in 19th century, he noticed that 80% of the wealth was owned by only 20% of the population. later,
- Pareto principle was developed by Juran in 1950. A Pareto chart is a special type of histogram that can easily be apply to find and prioritize quality problems, conditions, or their causes of in the organization (Juran and Godfrey, 1998).

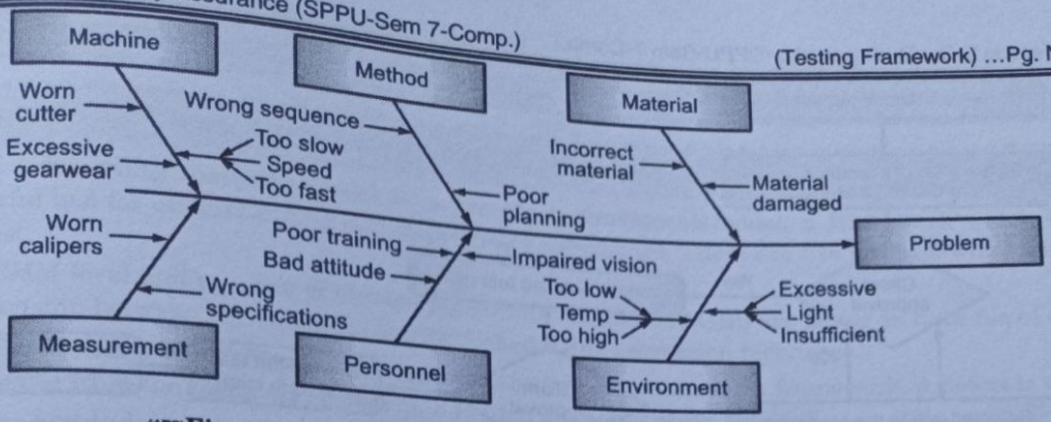
- On the other hand, it is a type of bar chart that shows the relative importance of variables, prioritized in descending order from left to right side of the chart.
- The aim of Pareto chart is to figure out the different kind of "nonconformity" from data figures, maintenance data, repair data, parts scrap rates, or other sources.
- Also, Pareto chart can generate a mean for investigating concerning quality improvement, and improving efficiency, "material waste, energy conservation, safety issues, cost reductions", etc., as
- Fig. 6.14.1 demonstrated concerning Pareto chart, it can able to improve the production before and after changes.



(1F4)**Fig. 6.14.1 : Pareto Charts**

6.15 FISHBONE DIAGRAM

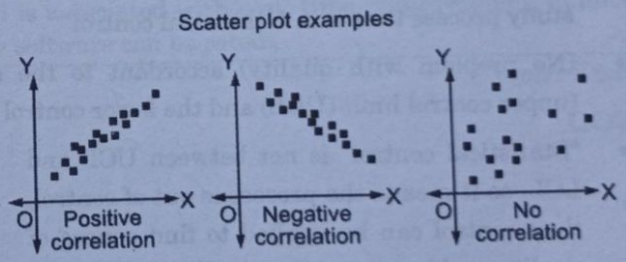
- Kaoru Ishikawa is considered by many researchers to be the founder and first promoter of the 'Fishbone' diagram (or Cause-and-Effect Diagram) for root cause analysis and the concept of Quality Control (QC) circles. Cause and effect diagram was developed by Dr. Kaoru Ishikawa in 1943.
- It has also two other names that are Ishikawa diagram and fishbone because the shape of the diagram looks like the skeleton of a fish to identify quality problems based on their degree of importance.
- The cause and effect diagram is a problem-solving tool that investigates and analizes systematically all the potential or real causes that result in a single effect.
- On the other hand, it is an efficient tool that equips the organization's management to explore for the possible causes of a problem.
- This diagram can provide the problem-solving efforts by "gathering and organizing the possible causes, reaching a common understanding of the problem, exposing gaps in existing knowledge, ranking the most probable causes, and studying each cause".
- The generic categories of the cause and effect diagram are usually six elements (causes) such as environment, materials, machine, measurement, man, and method, as indicated in Fig. 6.15.1. Furthermore, "potential causes" can be indicated by arrows entering the main cause arrow.



(1F5)**Fig. 6.15.1 : The cause and effect diagram (Fishbone Diagram)**

6.16 SCATTER DIAGRAM

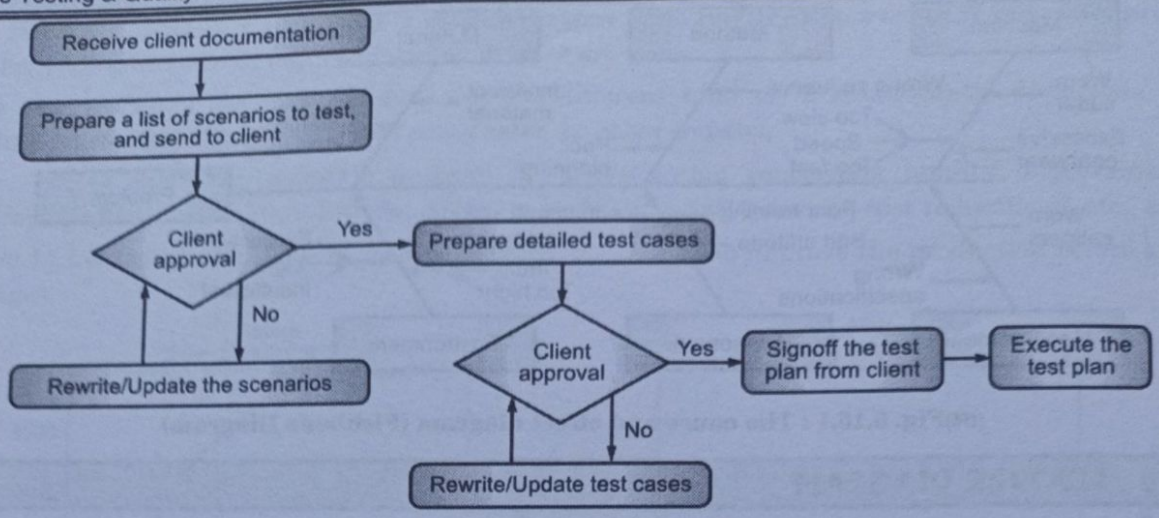
- Scatter diagram is a powerful tool to draw the distribution of information in two dimensions, which helps to detect and analyze a pattern relationships between two quality and compliance variables (as an independent variable and a dependent variable), and understanding if there is a relationship between them, so what kind of the relationship is (Weak or strong and positive or negative).
- The shape of the scatter diagram often shows the degree and direction of relationship between two variables, and the correlation may reveale the causes of a problem. Scatter diagrams are very useful in regression modeling.
- The scatter diagram can indicate that there is which one of these following correlation between two variables :
  (a) Positive correlation;
  (b) Negative correlation, and
  (c) No correlation, as demonstrated in Fig. 6.16.1. an operation, so it is very useful to find and improve quality into process.



(1F6)**Fig. 6.16.1 : Scatter Diagram**

6.17 FLOWCHART

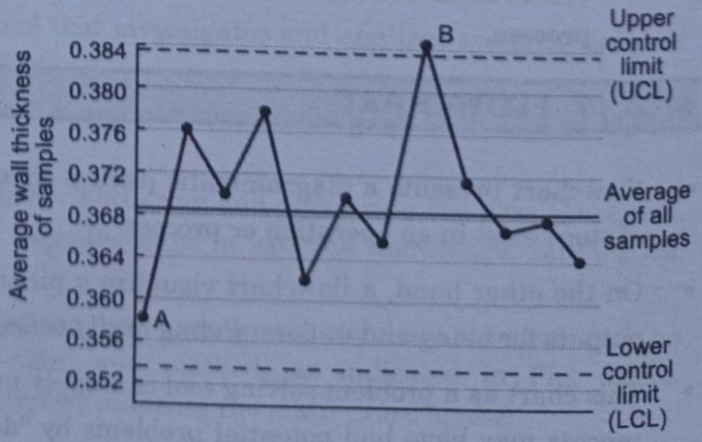
- Flowchart presents a diagrammatic picture that indicates a series of symbols to describe the sequence of steps exist in an operation or process.
- On the other hand, a flowchart visualize a picture including the inputs, activities, decision points, and outputs for using and understanding easily concerning the overall objective through process.
- This chart as a problem solving tool can apply methodically to detect and analyze the areas or points of process may have had potential problems by "documenting" and explaining an operation, so it is very useful to find and improve quality into process.



(1F7)**Fig. 6.17.1 : Flow Char Review Process**

6.18 CONTROL CHART

- Control chart or Shewhart control chart was introduced and developed by Walter A. Shewhart in the 1920s at the Bell Telephone Laboratories, and is likely the most "technically sophisticated" for quality management.
- Control charts are a special form of "run chart that it illustrates the amount and nature of variation in the process over time". Also, it can draw and describe what has been happening in the process.
- Therefore, it is very important to apply control chart, because it can observe and monitor process to study process that is in "statistical control".
- (No problem with quality) accordant to the samplings or samplings are between UCL and LCL (upper control limit (UCL) and the lower control limit (LCL)).
- "Statistical control" is not between UCL and LCL, so it means the process is out of control, then control can be applied to find causes of quality problem, as shown in Fig. 6.18.1 that A point is in control and B point is out of control. In addition, this chart can be utilized for estimating "the parameters" and "reducing the variability" in a process.
- The main aim of control chart is to prevent the defects in process. It is very essentiality for different businesses and industries, the reason is that unsatisfactory products or services are more coasted than spending expenses of prevention by some tools like control charts.



(1F8)**Fig. 6.18.1 : The Shewhart control chart**

6.19 DEFECT REMOVAL EFFECTIVENESS AND PROCESS MATURITY

(Testing Framework) ...Pg. No. (6-23)

UQ. Explain in brief Defect Removal Effectiveness and Process Maturity ? **SPPU - May 18 (Endsem)**

- CMM - capability maturity model is a process improvement model, a framework which is used as powerful tool for understanding and improving performance. There are five levels of which Level fifth is highest.
- The CMM level and number of defects present at various STAGES are based on data file of defect. The relationship between the variables is established using regression technique.
- CMM - capability maturity model is a process improvement model, a framework. A defect is nothing but a variance from the given specification, a hidden or coding error.
- Defects are undesirable, they cause increase in risk, revenue loss to the customer if they remain in the final product.
- Rectification may prove costly and results in risk and loss to the agency too. This is persistent from the innumerable surveys conducted by well recognized agencies.
- Defects are brought to the notice of project team by a process known as bug reporting. Defect reports are used to alert software programmers about the defect and gives them sufficient information to find root cause of problem and fix it. It provides information to technical writers and add test cases in the regression suite for next release. In this paper the relationship between CMM level and number of defect present in various STAGES like Requirement (REQ), Design (DSN), Coding (CDE), Document (DOC) is established.
- The relationship between CMM level and number of defects present in various STAGES like Requirement (REQ), Design (DSN), Coding (CDE), Document (DOC) are established in terms of prediction. Number of defects predicted are based on CMM level i.e. 0, 1, 3, 5.
- The number of defects present are associated with cost, time and quality. The lesser the number of defect indicate better the quality. Thus CMM level is associated with cost, time, improvement, quality and reduce rework. Based on it good or poor quality software can be rated.

...Chapter Ends

❑❑❑