US-SIT-403 Software Testing and Quality Assurance

Semester 4 : Unit 2

Dr Rakhi Gupta

2.1 Fundamentals of Testing

- Introduction, Necessity of testing, What is testing?
- Fundamental test process, The psychology of testing, Historical Perspective of Testing,
- Definitions of Testing, Approaches to Testing, Testing During Development Life Cycle,
- Requirement Traceability Matrix, Essentials of Software Testing, Workbench,
- Important Features of Testing Process, Misconceptions About Testing.

Introduction

- Testing

  The execution of a program to find its faults
- Verification

  The process of proving the programs correctness.
- Validation

  The process of finding errors by executing the program in a real environment
- Debugging

  Diagnosing the error and correct it

Most Common Software problems

- Incorrect calculation
- Incorrect data edits & ineffective data edits
- Incorrect matching and merging of data
- Data searches that yields incorrect results
- Incorrect processing of data relationship
- Incorrect coding / implementation of business rules
- Inadequate software performance

- Confusing or misleading data
- Software usability by end users &
- Obsolete Software
- Inconsistent processing
- Unreliable results or performance
- Inadequate support of business needs
- Incorrect or inadequate interfaces
- with other systems
- Inadequate performance and security controls
- Incorrect file handling

Necessity of testing

- The main aim of testing is to maintain the quality of the product.
- Testing has its own cycle and in each phase, all focus revolves around quality only.
- Software testing is the more cost-effective approach of handling software applications.
- Testing alleviates the need for a constant cycle of upgrades and fixes, as software testers identify bugs and errors before any such problems can arise.

Software testing is a process used to identify the correctness, completeness, and quality of developed computer software.

Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Testing is done in different forms at every phase of SDLC:

During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.

Reviewing the design in the design phase with the intent to improve the design is also considered as testing.

Objectives of testing

- Executing a program with the intent of finding an *error*.
- To check if the system meets the requirements and be executed successfully in the Intended environment.
- To check if the system is “ Fit for purpose”.
- To check if the system does what it is expected to do.

A Good Test Case

- A good test case is one that has a probability of finding an as yet undiscovered error.
- A successful test is one that uncovers a yet undiscovered error.
- A good test is not redundant.
- A good test should be “best of breed”.
- A good test should neither be too simple nor too complex.

What is testing

- **Software Testing**
- Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.
- It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest.
- The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

TESTABILITY

- Operability
- Observe-ability
- Controllability
- Decomposability
- Stability
- Understandability

QUALITY PRINCIPLES

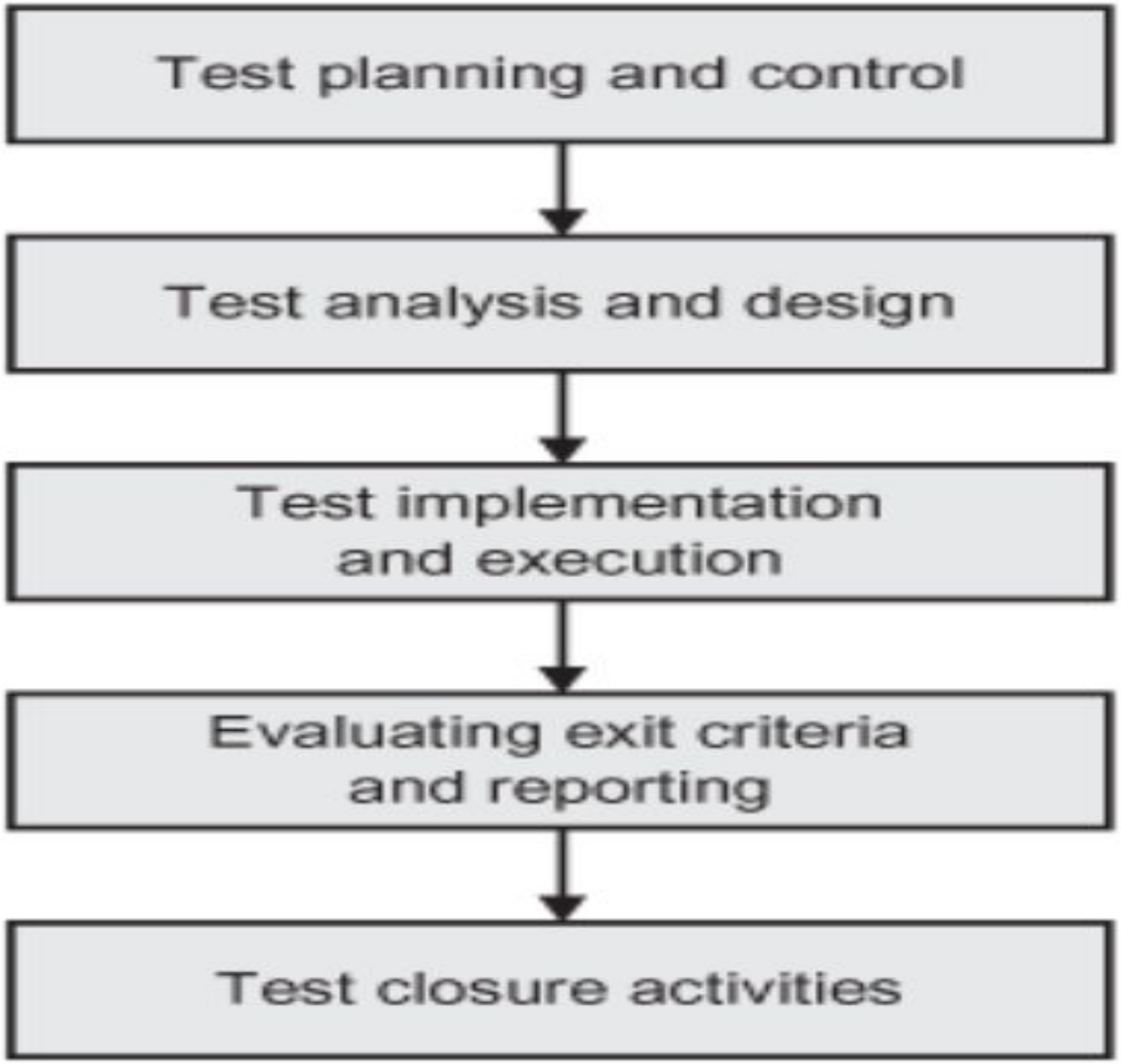
Quality - the most important factor affecting an organization's long-term performance.

Quality - the way to achieve improved productivity and competitiveness in any organization.

Quality - saves. It does not cost.

Quality - is the solution to the problem, not a problem.

Fundamental test process



Fundamental test process

- Testing is a process rather than a single activity. This process starts from test planning then designing test cases, preparing for execution and evaluating status till the test closure.
- We can divide the activities within the fundamental test process into the following basic steps:

1) Planning and Control
2) Analysis and Design
3) Implementation and Execution
4) Evaluating exit criteria and Reporting
5) Test Closure activities

1) Planning and Control

Test planning has following major tasks:

1. To determine the scope and risks and identify the objectives of testing.
2. To determine the test approach.
3. To implement the test policy and/or the test strategy. (Test strategy is an outline that describes the testing portion of the software development cycle. It is created to inform PM, testers and developers about some key issues of the testing process. This includes the testing objectives, method of testing, total time and resources required for the project and the testing environments.).
4. To determine the required test resources like people, test environments, PCs, etc.
5. To schedule test analysis and design tasks, test implementation, execution and evaluation.
6. To determine the Exit criteria we need to set criteria such as Coverage criteria. (Coverage criteria are the percentage of statements in the software that must be executed during testing. This will help us track whether we are completing test activities correctly. They will show us which tasks and checks we must complete for a particular level of testing before we can say that testing is finished.)

Test control

- **Test control** has the following major tasks:

i. To measure and analyze the results of reviews and testing.
ii. To monitor and document progress, test coverage and exit criteria.
iii. To provide information on testing.
iv. To initiate corrective actions.
v. To make decisions.

2) Analysis and Design

- **Test analysis and Test Design** has the following major tasks:

i. To review the test basis. (The test basis is the information we need in order to start the test analysis and create our own test cases. Basically it's a documentation on which test cases are based, such as requirements, design specifications, product risk analysis, architecture and interfaces. We can use the test basis documents to understand what the system should do once built.)
ii. To identify test conditions.
iii. To design the tests.
iv. To evaluate testability of the requirements and system.
v. To design the test environment set-up and identify and required infrastructure and tools.

3) Implementation and Execution:

- During test implementation and execution, we take the test conditions into test cases and procedures and other testware such as scripts for automation, the test environment and any other test infrastructure.
- Test cases : is a set of conditions under which a tester will determine whether an application is working correctly or not.
- Testware : is a term for all utilities that serve in combination for testing a software like scripts, the test environment and any other test infrastructure for later reuse.

Test implementation

- **Test implementation** has the following major task:
  i. To develop and prioritize our test cases by using techniques and create **test data** for those tests. (In order to test a software application you need to enter some data for testing most of the features. Any such specifically identified data which is used in tests is known as test data.)

- We also write some instructions for carrying out the tests which is known as **test procedures.**
  We may also need to automate some tests using **test harness** and automated tests scripts. (A test harness is a collection of software and test data for testing a program unit by running it under different conditions and monitoring its behavior and outputs.)
  ii. To create test suites from the test cases for efficient test execution.
  (Test suite is a collection of test cases that are used to test a software program to show that it has some specified set of behaviours. A test suite often contains detailed instructions and information for each collection of test cases on the system configuration to be used during testing. Test suites are used to group similar test cases together.)
  iii. To implement and verify the environment.

Test execution

- **Test execution** has the following major task:
  i. To execute test suites and individual test cases following the test procedures.
  ii. To re-execute the tests that previously failed in order to confirm a fix. This is known as **confirmation testing or re-testing.**
  iii. To log the outcome of the test execution and record the identities and versions of the software under tests. The **test log** is used for the audit trial. (A test log is nothing but, what are the test cases that we executed, in what order we executed, who executed that test cases and what is the status of the test case (pass/fail). These descriptions are documented and called as test log.).
  iv. To Compare actual results with expected results.
  v. Where there are differences between actual and expected results, it report discrepancies as Incidents.

Evaluating Exit criteria and Reporting:

- **) Evaluating Exit criteria and Reporting:**
  Based on the risk assessment of the project we will set the criteria for each test level against which we will measure the “enough testing”. These criteria vary from project to project and are known as **exit criteria**.
  Exit criteria come into picture, when:
  — Maximum test cases are executed with certain pass percentage.
  — Bug rate falls below certain level.
  — When achieved the deadlines.
- **Evaluating exit criteria** has the following major tasks:
  i. To check the test logs against the exit criteria specified in test planning.
  ii. To assess if more test are needed or if the exit criteria specified should be changed.
  iii. To write a test summary report for stakeholders.

5) Test Closure activities

- **5) Test Closure activities:**
  Test closure activities are done when software is delivered. The testing can be closed for the other reasons also like:
- When all the information has been gathered which are needed for the testing.
- When a project is cancelled.
- When some target is achieved.
- When a maintenance release or update is done.
- **Test closure activities** have the following major tasks:
  i. To check which planned deliverables are actually delivered and to ensure that all incident reports have been resolved.
  ii. To finalize and archive testware such as scripts, test environments, etc. for later reuse.
  iii. To handover the testware to the maintenance organization. They will give support to the software.
  iv To evaluate how the testing went and learn lessons for future releases and projects.

The psychology of testing

1. The comparison of the mindset of the tester and the developer.
2. The balance between self-testing and independent testing.
3. There should be clear and courteous communication and feedback on defects between tester and developer.

1. Comparison of the mindset of the tester and developer

- The testing and reviewing of the applications are different from the analysing and developing of it.
- If we are building or developing applications we are working positively to solve the problems during the development process and to make the product according to the user specification.
- However while testing or reviewing a product we are looking for the defects or failures in the product.
- Thus building the software requires a different mindset from testing the software.

2. The balance between self-testing and independent testing

- The comparison made on the mindset of the tester and the developer in the above article is just to compare the two different perspectives.
- It does not mean that the tester cannot be the programmer, or that the programmer cannot be the tester, although they often are separate roles. In fact programmers are the testers.
- They always test their component which they built. While testing their own code they find many problems so the programmers, architect and the developers always test their own code before giving it to anyone. However we all know that it is difficult to find our own mistakes.
- So, programmers, architect, business analyst depend on others to help test their work. This other person might be some other developer from the same team or the Testing specialists or professional testers.
- Giving applications to the testing specialists or professional testers allows an independent test of the system.
- **This degree of independence avoids author bias and is often more effective at finding defects and failures.**
- There is several level of independence in software testing which is listed here from the lowest level of independence to the highest:
  i. Tests by the person who wrote the item.
  ii. Tests by another person within the same team, like another programmer.
  iii. Tests by the person from some different group such as an independent test team.
  iv. Tests by a person from a different organization or company, such as outsourced testing or certification by an external body.

3. Clear and courteous communication and feedback on defects between tester and developer

- We all make mistakes and we sometimes get annoyed and upset or depressed when someone points them out. So, when as testers we run a test which is a good test from our viewpoint because we found the defects and failures in the software.
- But at the same time we need to be very careful as how we react or report the defects and failures to the programmers. We are pleased because we found a good bug but how will the requirement analyst, the designer, developer, project manager and customer react.
- The people who build the application may react defensively and take this reported defect as personal criticism.
- The project manager may be annoyed with everyone for holding up the project.
- The customer may lose confidence in the product because he can see defects.
- Because testing can be seen as destructive activity we need to take care while reporting our defects and failures as objectively and politely as possible.
- The balance between self-testing and independent testing

Historical perspective of testing

1. Debugging oriented testing
2. Demonstration oriented testing
3. Destruction oriented testing
4. Evaluation oriented testing
5. Prevention oriented testing

Traditional definition of testing

- Testing is done to establishing confidence that the program does what it is supposed to do
- Any activity aimed at evaluating an attribute or capability of a program or system with respect to user requirements
- Process of demonstrating that errors are not present
- Testing gives number of defects present which indirectly gives a measurement of software quality
- Testing is done to evaluate the program or system used for making software
- Testing is used to confirm that a program performs it's intended functions correctly

What is testing?

- An activity of identification of the differences between expected results and actual results
- Process of executing a program with the intention to find defects
- Detecting specification related errors and deviations of application working from specifications
- Establish confidence that a program does what it is supposed to do
- Any activity aimed at evaluating an attribute of a program or software
- Measurement of software quality in terms of coverage of testing in terms of requirements, functionality, features, number of defects found etc can give information about confidence level imparted to customer

What is testing?

- Process of evaluating processes used in software development
- Verifying that system satisfies it's specified requirements as defined and is fit for normal use
- Confirming that program performs it's intended functions correctly
- Process of operating a system or component under specified conditions, observing and recording the results of such processing and making evaluation of some aspect of system
- Process of analyzing a software item to detect the difference between existing and required conditions

Why testing is necessary

1. Understanding of customer requirements may differ from person to person
2. Development people assume that whatever they have developed is as per customer requirements
3. Different entities are involved in different phases of software development
4. Integration issues can be present
5. There is a possibility of blindfold

Approaches to Testing

- BIG Bang approach to testing
- Total Quality Management Approach

BIG Bang approach to testing

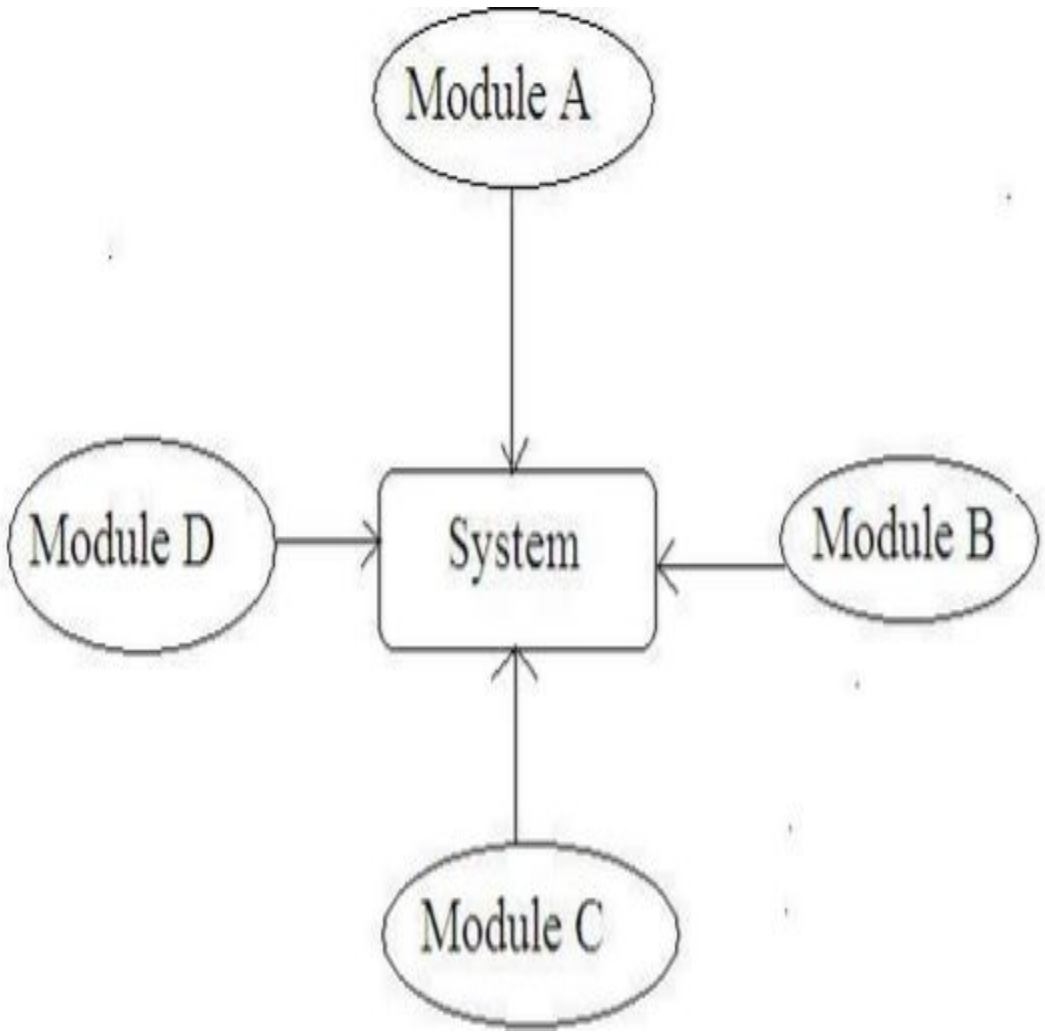
- In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole.

- In this approach individual modules are not integrated until and unless all the modules are ready.
- In Big Bang integration testing all the modules are integrated without performing any integration testing and then it's executed to know whether all the integrated modules are working fine or not.
- This approach is generally executed by those developers who follows the 'Run it and see' approach.
- Because of integrating everything at one time if any failures occurs then it become very difficult for the programmers to know the root cause of that failure.
- In case any bug arises then the developers has to detach the integrated modules in order to find the actual cause of the bug.

BIG Bang approach to testing

Suppose a system consists of four modules as displayed in the diagram above. In big bang integration all the four modules ‘Module A, Module B, Module C and Module D’ are integrated simultaneously and then the testing is performed.

Hence in this approach no individual integration testing is performed because of which the chances of critical failures increases.



Characteristics of Big bang approach

- Testing is a last phase of development life cycle when everything is finalized
- Characterized by huge rework, retesting, scrap, sorting
- Regression testing reveals many issues as correction may not be correct
- All requirements and designs cannot be covered in testing
- Major part of software build never gets tested

Advantage & Disadvantages of Big Bang Integration

- **Advantage of Big Bang Integration:**
- Big Bang testing has the advantage that everything is finished before integration testing starts.
- **Disadvantages of Big Bang Integration:**
- The major disadvantage is that in general it is very time consuming
- It is very difficult to trace the cause of failures because of this late integration.
- The chances of having critical failures are more because of integrating all the components together at same time.
- If any bug is found then it is very difficult to detach all the modules in order to find out the root cause of it.
- There is high probability of occurrence of the critical bugs in the production environment

Total Quality Management Approach

- TQM can be defined as a management technique for improving processes, products, services and the other approaches associated with the product.
- It focusses on the entire business and NOT just on a particular project or process.

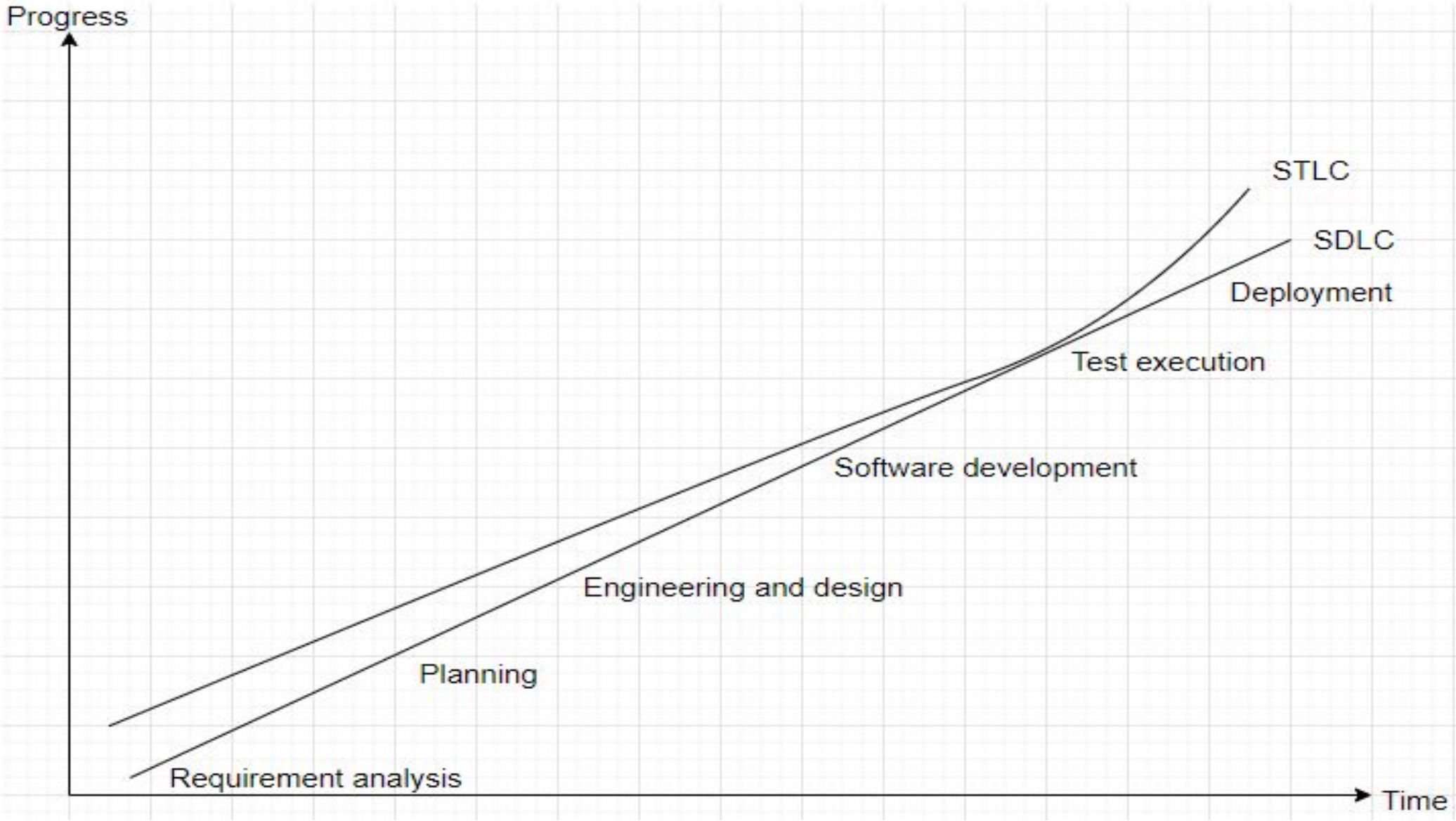
- TQM in cost perspective

1. Green money / cost of prevention
2. Blue money / cost of appraisal
3. Red money / cost of failure

Testing During Development Life Cycle

- **The role of the STLC in the SDLC**
- The Software Testing Life Cycle and the Software Development Life Cycle are closely related to each other, but they simultaneously pursue different tasks of the same goal, namely:
- collecting the requirements in the desired form and developing the declared functionality (as for the SDLC);
- analyzing requirements, assisting the client and the development team, and confirming the quality of the implemented functionality (as for the STLC).
- The overall goal is client satisfaction and achieving the highest possible score at the Verification and Validation stages.

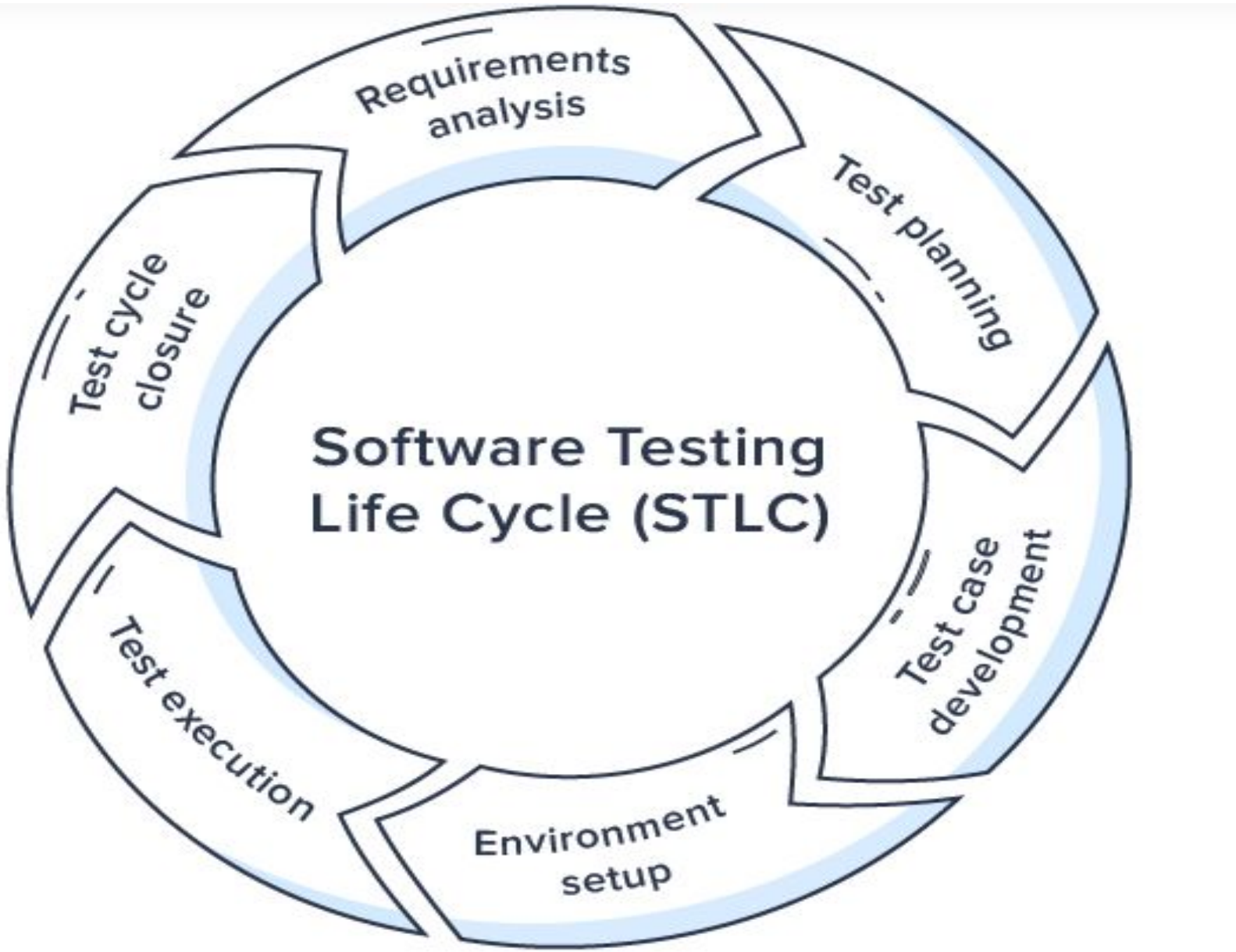
The role of the STLC in the SDLC can be represented with a double line graph:



STLC phases

- The STLC has several interconnected phases and is generally very similar to the SDLC system. These phases are sequential and are called:

1. Requirement Analysis;
2. Test Planning;
3. Test Case Development;
4. Test Environment Setup;
5. Test Execution;
6. Test Cycle Closure.


Requirements analysis
Test planning
Test case development
Environment setup
Test execution
Test cycle closure
Software Testing Life Cycle (STLC)

1. **Requirement Analysis**

- Requirements analysis is one of the most important phases because this is when you can fix project flaws almost completely free of charge. The requirements analysis phase also identifies the potential need for automated testing and allows making economic calculations of labor costs based on the project estimation. This is also when the Entry Criteria and the Exit Criteria are discussed and documented.

2. **Test Planning**

- At this phase, a test plan is formed. This is the concretization of all phases of the testing itself, timing, participants, and responsibilities. As a result of this, we receive data on:
- the participants and their roles in testing;
- the necessary testing tools;
- the necessary test environment.

3. Test Case Development

- Test development implies using manual and automated testing to achieve full coverage of the software's functionality, with the process being based on the requirements set up beforehand. More often than not, the test cases for automated testing are written separately, since the cases for manual testing are described in the form of cheat sheets.

4. Test Environment Setup

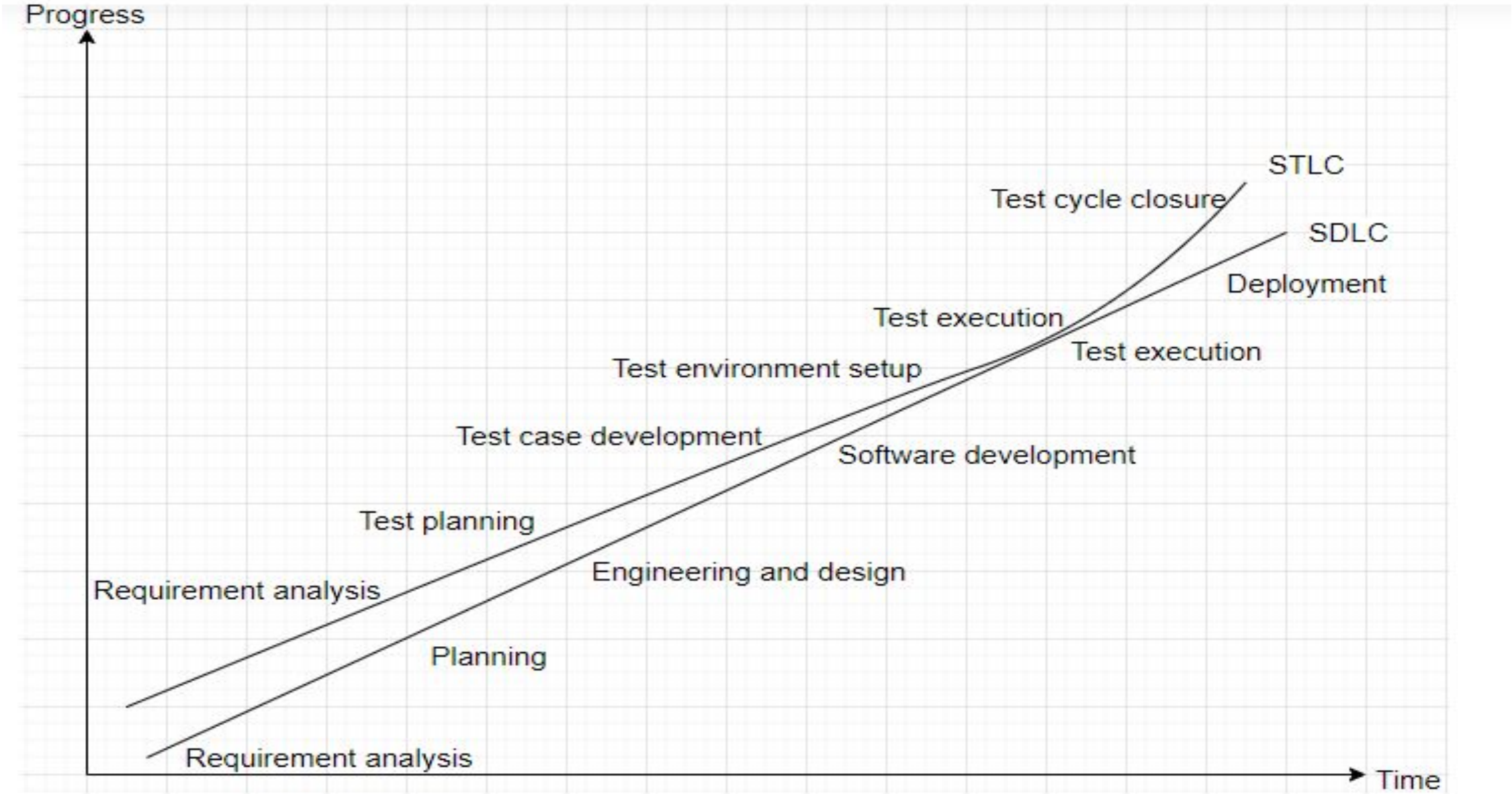
- The test plan makes it clear which test environment should be used. At this STLC phase, operating systems and virtual machines are configured, testing tools like Selenium, Katalon Studio, and the project’s test environment and databases are deployed. We also make requests to DevOps and Admins if support is needed.

5. Test Execution

- Tests are performed based on ready-made test documentation and a correctly configured test environment. All test results are recorded in the Test Management System. Negatively passed tests, where the actual result differs from the one expected, are recorded as errors and transferred to the development team for revision with subsequent rechecking after correction.

6. Test Cycle Closure

- The last phase of the STLC is the final generation of testing reports for the client. These should include the time spent, the percentage of errors found to positive test results, the total number of errors found and fixed. As far as the testing department is concerned, this is the moment for analyzing its work, summing up the results, analyzing its productivity, and the opportunity to make suggestions for improving the quality of testing.


Progress
STLC
Test cycle closure
SDLC
Deployment
Test execution
Test execution
Test environment setup
Test case development
Software development
Test planning
Engineering and design
Requirement analysis
Planning
Requirement analysis
Time

Manager's view of a software testing

- The product must be safe and reliable for use
- The product will work under normal and adverse conditions
- The product is meeting exactly what user wants
- The processes used for development and testing are capable of finding the defects

Tester's view of software testing

- Purpose of testing is to discover defects in the product and the process related to development and testing
- Process of trying to discover every conceivable fault or weakness in a work product

Customer's view of software testing

- Testing must be able to find all the possible defects in the software and related documentation
- Testing must give a confidence that software users are protected from any unreasonable failure
- Testing must ensure that any legal or regulatory requirements are complied during development

Requirement Traceability Matrix

- **What is Traceability Matrix? (TM)**
- A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.
- It is used to track the requirements and to check the current project requirements are met.
- **What is Requirement Traceability Matrix?**
- **Requirement Traceability Matrix (RTM)** is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software developement life cycle.
- The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

Why RTM is Important?

- The main agenda of every tester should be to understand the client's requirement and make sure that the output product should be defect-free. To achieve this goal, every QA should understand the requirement thoroughly and create positive and negative test cases.
- This would mean that the software requirements provided by the client have to be further split into different scenarios and further to test cases. Each of this case has to be executed individually.
- A question arises here on how to make sure that the requirement is tested considering all possible scenarios/cases? How to ensure that any requirement is not left out of the testing cycle?
- A simple way is to trace the requirement with its corresponding test scenarios and test cases. This merely is termed as 'Requirement Traceability Matrix.'
- The traceability matrix is typically a worksheet that contains the requirements with its all possible test scenarios and cases and their current state, i.e. if they have been passed or failed. This would help the testing team to understand the level of testing activities done for the specific product.

Which Parameters to include in Requirement Traceability Matrix?

- Requirement ID
- Requirement Type and Description
- Test Cases with Status

Req No	Req Desc	Testcase ID	Status
123	Login to the application	TC01,TC02,TC03	TC01-Pass TC02-Pass
345	Ticket Creation	TC04,TC05,TC06, TC07,TC08,TC09 TC010	TC04-Pass TC05-Pass TC06-Pass TC06-Fail TC07-No Run
456	Search Ticket	TC011,TC012, TC013,TC014	TC011-Pass TC012-Fail TC013-Pass TC014-No Run

Sample Requirement Traceability Matrix

Advantages of Requirement traceability Matrix

- Requirement traceability matrix can:

1. Show the requirement coverage in the number of test cases
2. Design status as well as execution status for the specific test case
3. If there is any User Acceptance test to be done by the users, then UAT status can also be captured in the same matrix.
4. The related defects and the current state can also be mentioned in the same matrix.
5. It confirms 100% test coverage
6. It highlights any requirements missing or document inconsistencies
7. It shows the overall defects or execution status with a focus on business requirements
8. It helps in analyzing or estimating the impact on the QA team's work with respect to revisiting or re-working on the test cases

Types of Traceability Test Matrix

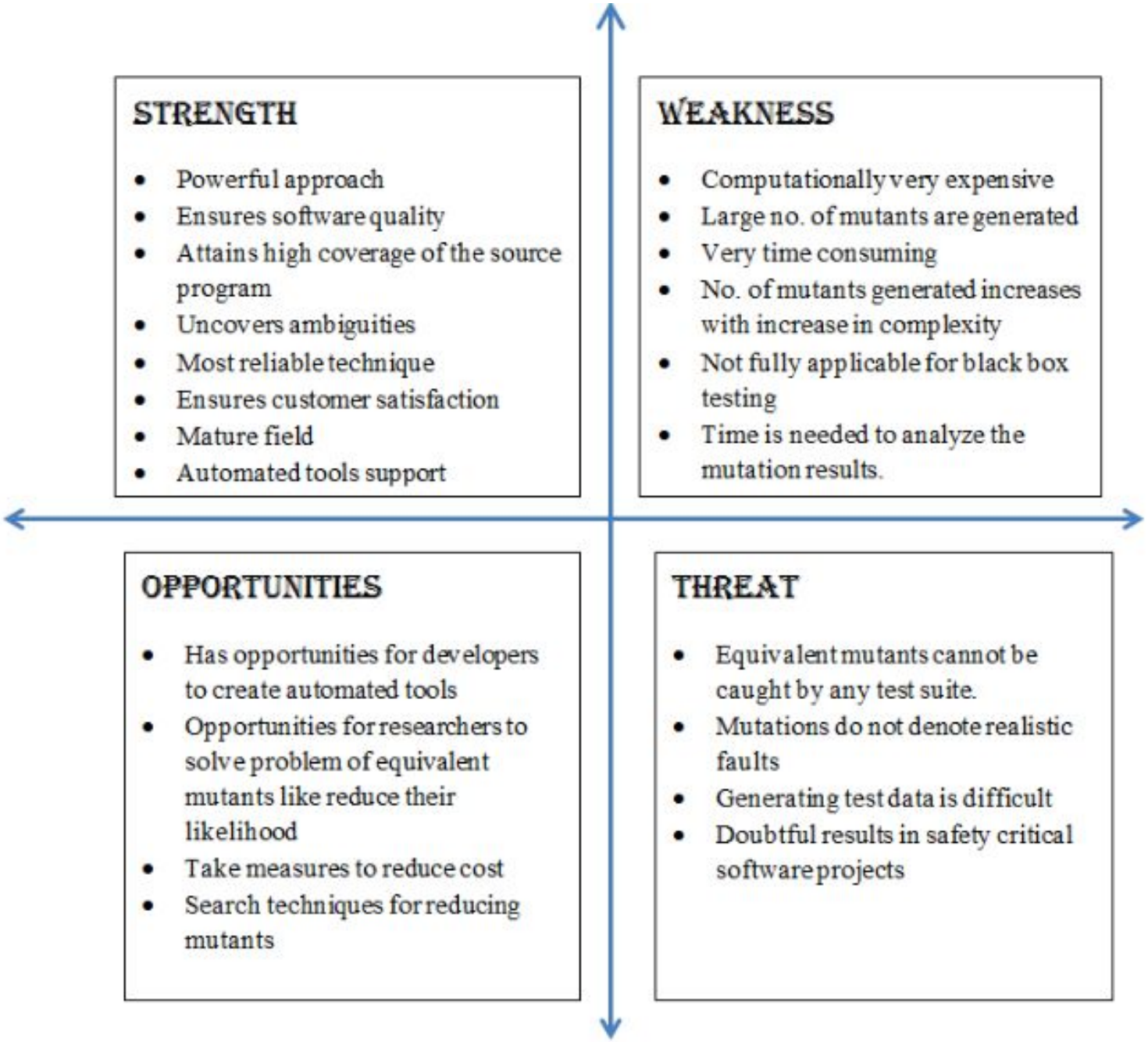
- In Software Engineering, traceability matrix can be divided into three major component as mentioned below:
- **Forward traceability**: This matrix is used to check whether the project progresses in the desired direction and for the right product. It makes sure that each requirement is applied to the product and that each requirement is tested thoroughly. It maps requirements to test cases.
- **Backward or reverse traceability:** It is used to ensure whether the current product remains on the right track. The purpose behind this type of traceability is to verify that we are not expanding the scope of the project by adding code, design elements, test or other work that is not specified in the requirements. It maps test cases to requirements.
- **Bi-directional traceability (Forward+Backward):** This traceability matrix ensures that all requirements are covered by test cases. It analyzes the impact of a change in requirements affected by the Defect in a work product and vice versa.

Essentials of Software Testing

1. A **test strategy** that tells you what types of testing and the amount of testing you think will work best at finding the defects that are lurking in the software
2. A **testing plan** of the actual testing tasks you will need to execute to carry out that strategy
3. **Test cases** that have been prepared in advance in the form of detailed examples you will use to check that the software will actually meet its requirements
4. **Test data** consisting of both input test data and database test data to use while you are executing your test cases, and
5. A **test environment** which you will use to carry out your testing.

SWOT Analysis in Software Testing

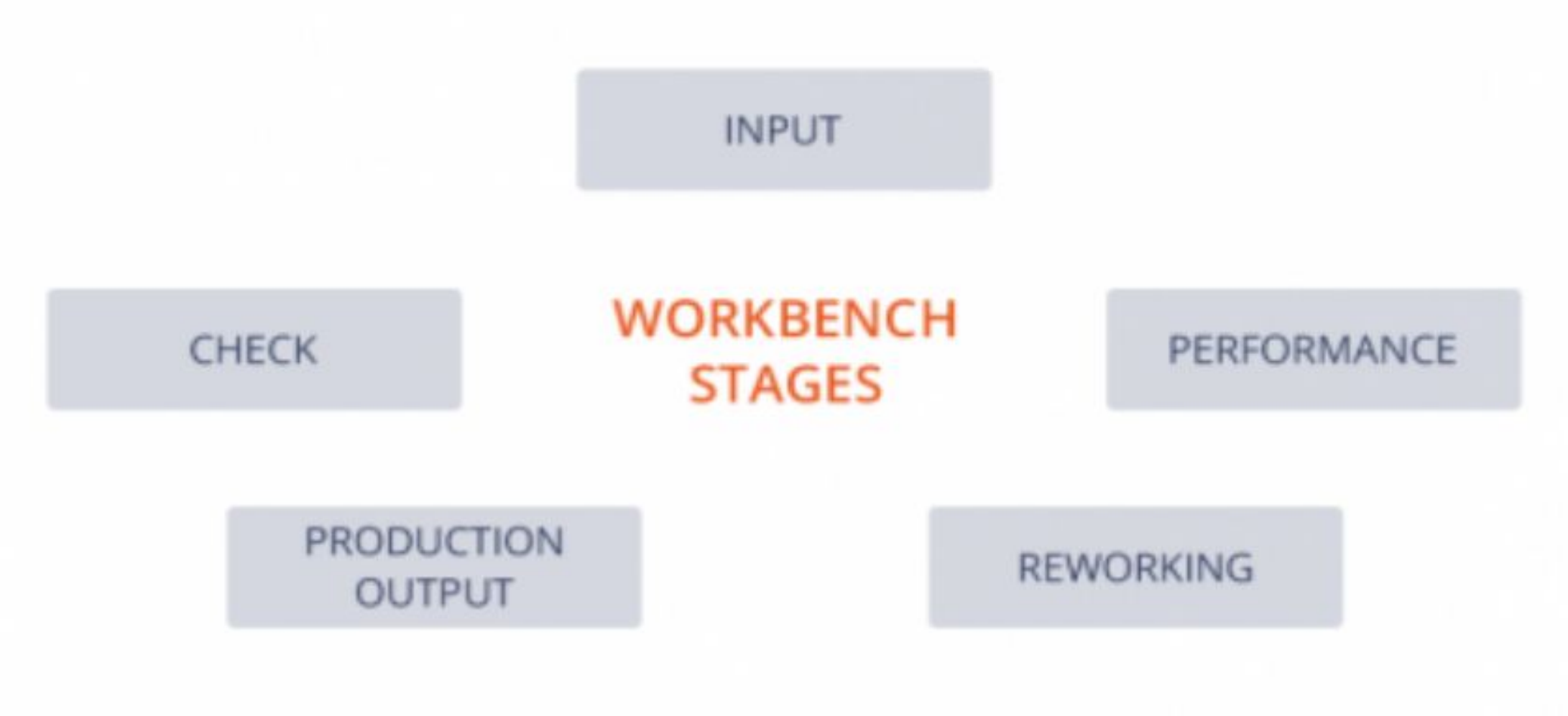
The SWOT (Strengths, Weaknesses, Opportunities, and Threats) analysis is presented giving an insight to the use of ontologies to enrich and enhance Software Engineering processes.



Testing Workbench

- **Workbench Concept: Goals and Stages**
- This is a method which aims to examine and verify the structure of testing performance by detailed documenting.
- Workbench process has its common stages and steps which serve for different test assignments. The common stages of each workbench include:

Testing Workbench



- **Input.** It is the initial workbench stage. Each certain assignment should contain its initial and outcome (input and output) requirements to know the available parameters and expected results. Each workbench has its specific inputs depending on the type of product under testing.
- **Performance.** The priority aim of the entire testing is in the transformation of the initial parameters to outcome requirements and reach the prescribed results.
- **Check.** It is an examination of output parameters after the performance phase to verify its accordance with the expected ones.
- **Production output.** It is the final stage of a workbench in case the check confirmed the properly conducted performance.
- **Reworking.** If the outcome parameters are not in compliance with the desired result, it is necessary to return to the performance phase and conduct it from the beginning.

Important Features of Testing Process

1. Validity: The first important characteristic of a good test is validity. The test must.
2. Reliability: A good test should be highly reliable. This means that the test should give.
3. Objectivity: By objectivity of a measuring instrument is meant for the degree to which.
4. Norms. : Norms are sets of score obtained by whom the test is intended. The scores obtained by these groups provide a basic for interpreting any individual score

Misconceptions About Testing

- **Misconception 1: No special skills are required for testing**
- The principle that drives such organizations is that ‘anyone can be put to testing’. However, testing is a specialized process and may not be efficiently performed by unskilled people, not wary of the different methodologies used in software testing.
- **Misconception 2: It is possible to test the system completely**
- Testing a system completely is not possible because there are too many combinations of data and user input and program paths, as well as hardware or software platforms on which the tested product runs, to allow complete testing in a reasonable period of time. Even if testing a system entirely could take place, it would be extremely expensive for the efforts to be equivalent to costs involved.
- **Misconception 3: You can have ‘zero defect’ or bug-free software**
- A general rule in testing is that there is simply not enough time and money to test everything. The goal is to test well-enough to deliver ‘good enough’ software quality. Thus, priority or importance of the tests plays a very important role in determining which tests should be preferred over the others.
- In addition, no matter how good your software may be, it runs on an operating system with bugs, with device drivers that have bugs, on computer hardware that has bugs, and so on. The bugs in the environment on which your software runs can also cause problems in your software. Further, since there exist a virtually infinite variety of computer hardware and operating systems (and their associated bugs) on which your software may be run by your customers, there is no way to test them all.
- A zero-defect software is, therefore, a myth. If your software could deliver a reliable performance without serious bugs affecting its usability, the objective of software testing is complete.

- **Misconception 4: Software Testing is exclusively a quality control activity**
- Actually, software testing is partly a combination of both quality assurance (QA) and quality control (QC) activity. Quality control refers to measurement or inspection activities that are intended to compare the actual results with the expected results of a project. This is only a subset of software testing, as QC rarely takes the customer's needs as a priority. Possible examples of QC-related testing include regression testing. In actuality, software testing is a process of exploration in which there is an iterative evolving process of test design, execution, and problem identification.
- **Misconception 5: Software testing slows us down**
- Product release schedules have been driven very hard by forces such as the need to be first to market, thereby putting extreme pressure on the speed of development. When project schedules slip, testing gets squeezed the most. There have been great advances in software development platforms and tools, both proprietary and available as shareware, open source or freeware, that significantly accelerate the development process. In addition, software development methodologies such as Agile and Extreme Programming speed up software development. However, traditional testing has not kept pace with these advances and has not experienced a corresponding increase in speed. This leads to the perception that testing slows us down, as it hasn't sped up as the development process has over the course of time.
- **Misconception 6: Software testing is nothing but software development**
- While software testing is a part of the overall software development process, it is actually a very separate function. It is common to use software development, software design and programming interchangeably. It is not, however, common to say 'software development' when you mean 'software testing'.
- Consider this illustration- Often, people think of sales and marketing together. The two words are often spoken together as if to describe a single function: the generation of revenue. However, these are two very different activities. Marketing spends money, which could be viewed as an indirect sales activity because it ultimately generates leads and opportunities that turn into revenue. Sales, however, makes money because it brings in booked revenue. Likewise, software development is a creative activity which results in the generation of an eventual product, which in turn will generate revenue. It is directly constructive. Testing, however, is a destructive activity in which problems in the product under development are brought to light. Testing costs money, yet the results can lead to delays in the product release and eventual revenue.

Misconceptions about Testing and Defects

- Testers can test the quality of product at end of development process.
- Defect found in testing are blamed on developers .
- Defect found by customers are blamed on testers.
- Complete testing is done
- Zero defect software (product) creation.
- Tester can find all defects in limited period
- Testing is manual process, Automation involve optionally
- Testing process is simple than development step.
- Testing required less efforts
- Quality assurance is secondary based on brand value.

2.2 Principles of Software Testing

- Salient Features of Good Testing, Test Policy, Test Strategy or Test Approach, Test Planning
- Testing Process and Number of Defects Found in Testing, Test Team Efficiency, Mutation Testing, Challenges in Testing, Test Team Approach
- Process Problems Faced by Testing, Cost Aspect of Testing, Establishing Testing Policy, Methods
- Structured Approach to Testing, Categories of Defect, Defect, Error, or Mistake in Software

Generalized Principles of Software testing,

1. Programmers/ team must avoid testing their own works products
2. Thoroughly inspect results of each test case has to found to identify weaker area in software
3. Defects indicates process failure
4. Initiate actions for correction, corrective action & preventive action.

Salient Features of Good Testing

1. Capturing User requirements
2. Capturing User Needs
3. Design Objectives
4. User Interface
5. Internal Structures
6. Execution of Code

Salient features of good Software testing

1. Capture user requirements
2. Capturing user needs
3. Design objective must define properly
4. User interface simple and easy to understand
5. Internal Structure complex but simple to understand
6. Execution of code reduce risk of failure

Test Policy

Test Policy
Organizational Level Document
Test Strategy
Program Level Document
Test Plan
Project Level Document
Test Procedures & Test Cases
Test Reports
Bug & Issue Reports

1. TEST POLICY

- Test policy is a document described at the organization level and gives the organizational insight for the test activities.
- It is determined by the senior management of the organization and defines the test principles that the organization has adopt. The test policy is very high-level document and at the top of the test documentation structure.
- Organizations may prefer to publish their test policy in a sentence, as well as a separate document. Also they may use this policy in both development and maintenance projects.
- The test policy shall describe the followings:

1. Clear answer to the question of “What does testing means for the organization”
2. Test objectives that the organization have
3. The definition of the testing process used by the organization to increase the quality of the software developed
4. How the organization will measure the effectiveness and efficiency of the test while achieving goals
5. How the organization will improve its test processes

Test Policies

- Test policies are generally defined by senior management try to cover all aspects of testing.
- It decides framework of testing towards customer satisfaction.
- Testing will performed against TQM principles to find and fulfill customer centric approach

2. TEST STRATEGY

- Test strategy document is prepared at the program level and includes general test strategy, management principles, processes and approaches for the tests to be performed for a software in detail.
- The test strategy document is also a high level document and is usually written by the test manager and the project manager in the top level organization. It is generally prepared in large scale projects and does not need much updating. In small scale projects, test strategies and test approach may be included in the test plan, and also the test strategy document may not be written separately.
- Test approach and test activities included in the test strategy document must be consistent with the test policies of the organization.

- The test strategy document may applicable for a program / system that contains multiple projects and describes;

1. Objective / scope of testing
2. In-scope / out of scope items for testing
3. Test levels (Unit, System, Integration, System Integration)
4. Test types (Functional / Non-Functional)
5. Entry / Exit / Stop / Resumption Criteria for testing (for different levels / phases)
6. Risks to be addressed
7. Test environment
8. Test case design methodology
9. Test methodology (Top-down / bottom-up / risk based)
10. Test control and reporting
11. Test automation approach
12. Test tools to be used
13. Defect management approach
14. Defect classification
15. Retesting & regression approach

Test Strategy or Approach

- Globally, there is unique test policy decided by management but it can varies as per customer, Deadline, Product, Project etc.
- Definition of coverage is defined for testing (Functional, requirements and features for different products, Customer and project)
- Level of testing must monitored and maintained (unit to acceptance testing)
- How much testing manually and when to automate has been planned and executed.
- Number of developer and testers are assigned in proportion.

Test Planning

- Test planning is first activity of test team
- Test plan are defined throughout SDLC
- Test plan must be realistic and talk about limitations and constraints in system

1. Plan testing efforts adequately with assumption that defects are exists in software
2. Successful tester founds defect in systems not appreciates developer
3. Testing dose not completed at end of development cycle, (plan for maintenance)
4. Verification(do-right thing) and validation(right-thing do) must done at each phase of testing.

3. TEST PLAN

- Test plan is a document prepared at the project level. In general it defines work products to be tested, how they will be tested (test cases) and test type distribution among testers.
- Test plan also includes test environment and test tools to be used during the project, the persons responsible for the tests and their responsibilities, test levels and test types, test schedule planned for test runs, and the principles of management and reporting of errors / bugs.
- Test plan is usually prepared by the test manager or test leader in the test organization and shared with the entire team in the project. It is a living document throughout the project and should be kept under revision control as it’s updated.
- The information in the test plan document must be consistent with the organization’s test policy and test strategy.

The test plan may describe the followings:

1. All test strategies specific to the project
2. Test estimations & test schedule
3. Test organization / roles / responsibilities
4. Test deliverables
5. Test reporting principles

IEEE Std 829 (IEEE Standard for Software Test Documentation

1. Test Plan Identifier (name, number, revision number of the document)
2. References
3. Introduction
4. Test Items
5. Software Risk Issues
6. Features to be Tested
7. Features not to Tested
8. Approach
9. Item Pass / Fail Criteria
10. Suspension Criteria and Resumption Requirements
11. Test Deliverables
12. Remaining Test Task
13. Environmental Needs
14. Staffing and training needs
15. Responsibilities
16. Schedule
17. Planning Risks and Contingencies
18. Approvals
19. Glossary

Test planning

- Plan testing efforts with an assumption that defects are there
- If defects are not found, it is failure of testing activity
- Successful Tester is not one who appreciates development but one who finds defect
- Testing is not a formality to be completed at the end of development cycle

Differences between verification and validation

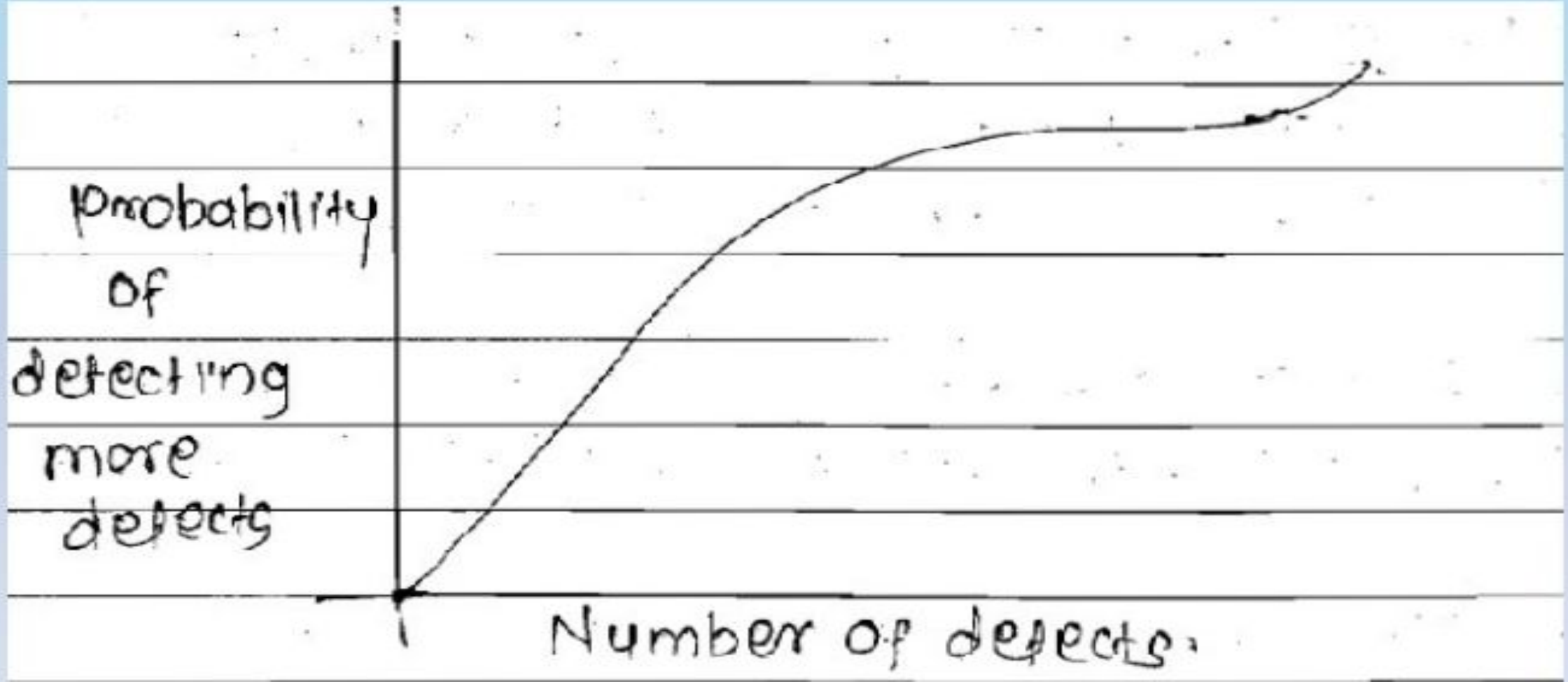
Verification	Validation
Verification is an activity where we check the work products with reference to standards, guidelines, procedures	Validation is an activity to find whether the software achieves whatever is defined by requirements
Verification is prevention based. It tries to check the process adherence.	Validation is detection based. It checks the product attributes.
Verification talks about a process, standard, guideline etc	Validation talks about the product
Verification is also termed as white box testing or static testing as the work product undergoes a review	Validation is also termed as black box testing or dynamic testing as workproduct is executed
Verification may be based on opinion of reviewer and may change from person to person	Validation is based on facts and is generally independent of a person

Verification can find about 60% of the defects	Validation can find about 30% of the defects
Verification involves • Reviews • Walkthroughs • Inspection • Audits	Validation involves all kinds of testing • System testing • User Interface testing • Stress testing etc
Verification can give • Statement coverage • Decision coverage • Path coverage	Validation can give • Requirement coverage • Feature coverage • Functionality coverage
Some parts of software can undergo verification only • Coding guidelines • Nesting • Commenting • Declaration of variables	Some characteristics of software can be proven by validation only • Stress testing • Performance testing • Volume testing • Security testing

Testing Process and number of defects found in testing

- In real time, As we find more defects, there is probability of finding some more defects. Governed by teams defect finding ability.
- Figure :- defect trend
- As tester assures one defect may leads multiple defects and count of defect will increasing

defect trend



Test team efficiency

- It's a defect finding rate of particular test team
- More the efficient team, Less the defects found in system by customer, As these are resolved by developer in iteration of product/service.
- Every test manager must aware of test team efficiency which should tends towards 100%
- Ex. 90% means, 500 defects are reported by team still there exists 50 more defects possible, 95% still 25 more. 100 no more defects(practically not achievable).
- Efficiency can calculate using old success rates.

Mutation testing

- This testing use to check capability of test program and test cases to find defects
- This also known as test case efficiency.
- If original program is changed(due to any reason) and some defects are added in system called mutant of original program and process termed as mutation.

Challenges in testing

1. Challenges associated with developer team, Customer needs, Management perspective
2. Requirements are not clear, complete, consistent, measurable and testable
3. Requirements are wrongly documented & inspected by business & system analyst
4. Code logic may difficult to capture
5. Error handling may be difficult to capture

Test team & approaches

- Type of organization & Type of product developed to be tested define **test team**.
- There may or may not be separate testing team for each product to be developed.

1. **Developer become testers** –Small size team
2. **Independent testing team-** Assigns test manager
3. **Domain experts doing software testing** –System and acceptance testing
4. **Test team reporting development manager**-Most common test team approach

Process problem faced in Testing

- Defects are introduce in software due to incorrect processes, The basic constituents (part) of processes are :-

1. People :- user-specifying requirements, Business analyst-document requirements, Test manager define test plan, tester define test case etc.
2. Material:- Requirement docs, development standards, guidelines must provide properly
3. Machines :-Simulators, real time objects defined
4. Methods:- Test planning, defining, test data may not proper.
5. Economics of testing:-Customer dissatisfaction inversely proportional to testing efforts

Cost aspect in Testing

- Cost of quality includes cost of appraisal, prevention and failure cost of product/service
- Testing is costly to organization, so it try to keep as minimum as possible
- As per economics of testing cost has to manage
- Cost include :-Capture requirements, Conduct analysis, ask queries, Design test cases high and low level, write code, automation test tool, create final product and maintain.

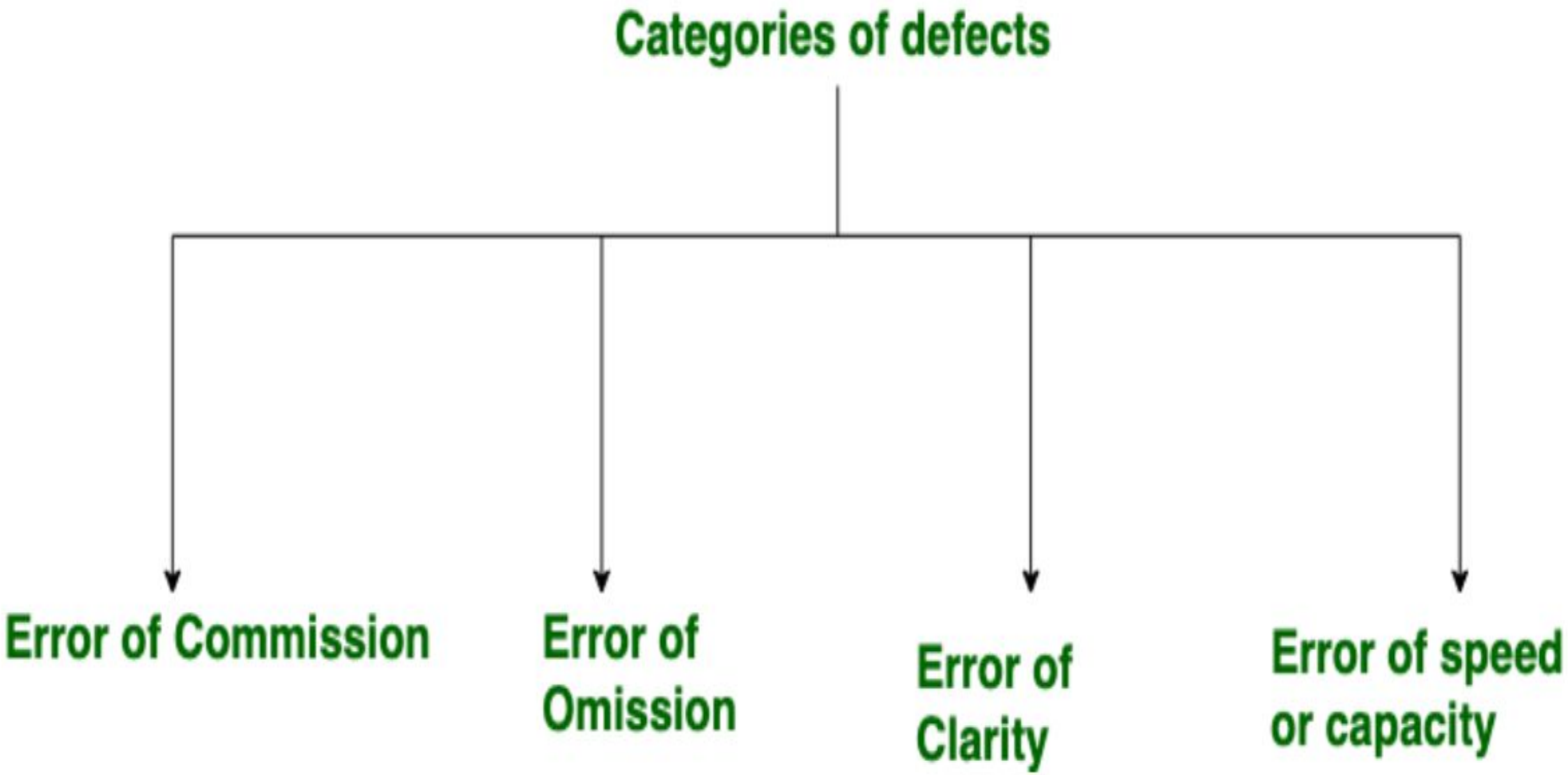
Structured approach for testing

- Establish testing policies
- Use standard methods
- Four types of wastes are involved in structured approch
- Waste in wrong development
- Waste in testing to detect defects
- Wastage as wrong specification

Categories of Defect

- **Software Defect** is some kind of error, flaw or some kind of mistake from the development team which prevent the software from the smooth working. It directly affect software quality, software quality is some thing how smooth and reliable your software is. Smoothness and reliability is how less defects your software have.
- **Categories of defects:**
  Categories of defects are:

1. Errors of commissions
2. Errors of omissions
3. Errors of clarity
4. Error of speed and capacity.


Categories of defects
Error of Commission
Error of Omission
Error of Clarity
Error of speed or capacity

1. **Error of Commission:**
   Commission means instruction or some kind of command given. Now the error in commission means the error in made in command or instruction. For example, suppose I wrote a loop which I was trying to run 10 times but I command it to run more than 10 times by mistake this is the error of commission.
2. **Errors of Omissions:**
   As name is already describing error of omission is some thing which happens accidentally. Omission word means something left out or executed. Practical most common example of this error is suppose we make a function in programming open its bracket but forget to close at the end.
3. **Error of Clarity:**
   The most common error in the natural languages. This error happens due to miss understanding between the developer and client. It travels most of the time from the requirements to the software.
4. **Error of Speed or Capacity:**
   The name of the error is itself enough i think to tell about it this error. Your software is working fine but not working in the required time this is the error of speed. When it comes to capacity it can be relevant to memory. For example, a small integer is declared where the long integer was required.

Defect, Error, or Mistake in Software

- "A mistake in coding is called Error, error found by tester is called **Defect**, defect accepted by development team then it is called Bug, build does not meet the requirements then it Is Failure."

- **What is a bug?**
- In software testing, a bug is the informal name of defects, which means that software or application is not working as per the requirement. When we have some coding error, it leads a program to its breakdown, which is known as **a bug**. The **test engineers** use the terminology **Bug**.
- If a **QA (Quality Analyst)** detect a bug, they can reproduce the bug and record it with the help of the **bug report template**.
- **What is a Defect?**
- When the application is not working as per the requirement is knows as **defects**. It is specified as the aberration from the **actual and expected result** of the application or software.
- In other words, we can say that the bug announced by the **programmer** and inside the code is called a **Defect.**
- **What is Error?**
- The Problem in code leads to errors, which means that a mistake can occur due to the developer's coding error as the developer misunderstood the requirement or the requirement was not defined correctly. The **developers** use the term **error**.

2.3 Testing Methodologies

- Developing Test Strategy: Developing Testing Methodologies (Test Plan), Testing Process, Attitude Towards Testing (Common People Issues)
- Test Methodologies/Approaches : Skills Required by Tester
- Testing throughout the software life cycle, Software development models
- Test levels, Test types, the targets of testing, Maintenance testing

Developing Test Strategy

- **What is a Test Strategy?**
- Test Strategy is a high-level document (static document) and usually developed by a project manager.
- It is a document which captures the approach on how we go about testing the product and achieve the goals.
- It is normally derived from the Business Requirement Specification (BRS).
- Documents like Test Plan are prepared by keeping this document as base.

Test Strategy vs Test Plan

Test Strategy	Test Plan
Test Strategy is a high level document which captures the approach on how we go about testing the product and achieve the goals.	Test plan document is a document which contains the plan for all the testing activities to be done to deliver a quality product.
Components of Test strategy includes Scope and overview, Test Approach, Testing tools, Industry standards to follow, Test deliverables, Testing metrics, Requirement Traceability Matrix, Risk and mitigation, Reporting tool, Test summary	Components of Test plan includes Test Plan Identifier, Features To Be Tested, Features Not To Be Tested, Approach, Pass/Fail Criteria, Suspension Criteria, Test Deliverables, Responsibilities, Staffing and Training Needs, Risks and Contingencies etc.,
It is developed by the project manager	It is prepared by test lead or test manager
It is derived from the Business requirement specifications (BRS)	It is derived from the Product Description, SRS, or Use Case documents
It is a static document. Once defined, it cannot be changed	It is a dynamic document. It can be changed
It is defined at organization level and can be used for other projects of similar nature	It is defined at project level

Key Components of Testing Strategy

1. Select Rank test factors for the given application
2. Identify System development phases and related test factors
3. Identify associated risks with each selected factor in case if it is not achieved
4. Identify phase in which risks of not meeting a test factor needs to be addressed

Table of Contents

1. When writing a test strategy, the following aspects should be considered:
2. Testing objectives
3. Testing guidelines
4. Testing approach
5. Roles and responsibilities
6. Levels of testing
7. Test requirements
8. Test deliverable
9. Entry and exit criteria
10. Defect management
11. What test reports will be provided
12. Test environment information and migration procedures
13. Test Constraints
14. Test Risk including project and product risks

Developing Testing Methodologies (Test Plan)

1. Acquire and study test strategy as defined
2. Determine the type of development project being executed
3. Determine the type of software system being made
4. Identify tactical risks related to development
5. Determine when testing must occur during life cycle
6. Steps to develop customized test strategy
7. Type of development methodology impacts test plan decisions .

Testing Process

- Defining Test Policy
- Defining Test strategy
- Preparing Test Plan
- Establishing Testing Objectives for be Achieved
- Designing Test Scenarios and Test cases
- Writing/ reviewing Test cases
- Defining Test Data
- Creation of test Bed
- Executing Test Cases
- Test results
- Test Analysis
- Performing Retesting
- Root Cause analysis

Attitude Towards Testing

- It starts from test policy and ends to test strategy definition
- Cross-cultural Differences
- Lack of proper communication
- Developers may not accept the defects in first place
- Conflict between developer and tester

Test methodologies

- Black box testing
- White box testing
- Grey box testing

Black box testing

- In black box testing product is tested as per software specifications or requirement statement defined by business analysts/system analysts/customer.
- It involves testing system/components considering inputs, outputs and general functionalities as defined in requirement specification
- Mainly talks about the requirement specification given by customer
- It does not consider any internal processing by the system
- This testing is with the view as if a user is testing the system.
- Black box functional testing is generally performed for integration testing, system testing, and acceptance testing
- It is mainly driven by the test data used during testing.

Advantages of black box testing

- It is the only method to prove that software dose what it is supposed to do and is dose not do something which can cause a problem to user/customer.
- It is the only method to show that software is living and it really works
- Performance an security testing can be done by black box testing methodology

Disadvantages of black box testing

- Some logical errors and internal structure problems in coding can be missed in black box testing
- If an application calls common function again and again, and it will be tested so many times that it leads redundant testing

White box testing

- White box testing is done on the basis of internal structures of software as defined by requirements, designs, coding standards and guidelines.
- It starts with reviews of requirements, designs and codes.
- Mainly deals with the structure and design of the software product.
- It requires that tester must have knowledge about development processes and artifacts including various platform, databases etc.
- Test case designing is based on how test artifacts are created and used during testing.

Advantages of White box testing

- It can check whether the coding standards, commenting reuse have been followed or not
- White box testing is the cost effective way of finding defects as it helps in reducing stage contamination.
- Code complexity, commenting styles and reuse can only be verified by white box testing methodologies.

Disadvantages of White box testing

- It does not ensure that user requirements are met correctly
- It does not establish whether decisions, conditions, paths and statements covered during reviews are sufficient or not for the given set of requirements.
- It is dominated by the use of checklists which may be retired directly in the work product and one must do a thorough analysis of all defects

Grey box testing

- Grey box testing is done on the basis of internal structures of software as defined by requirements, designs, coding standards and guidelines as well as the functional and non-functional requirement specification.
- It combines verification techniques with validation techniques where one can ensure that software is build correctly and also works.
- It talks mainly about a combination approaches, viz black and white box testing at the same.

Advantages of Grey box testing

- It checks whether the work product works in a correct manner, both functionally as well as structurally

Disadvantages of Grey box testing

- Knowledge of some automation tools along with their configuration is essential for performing grey box testing.

Raising Management Awareness for Testing

Management should be made aware of the importance of testing a product because the funds required will be provided by them.

Without adequate funding for testing, software quality suffers.

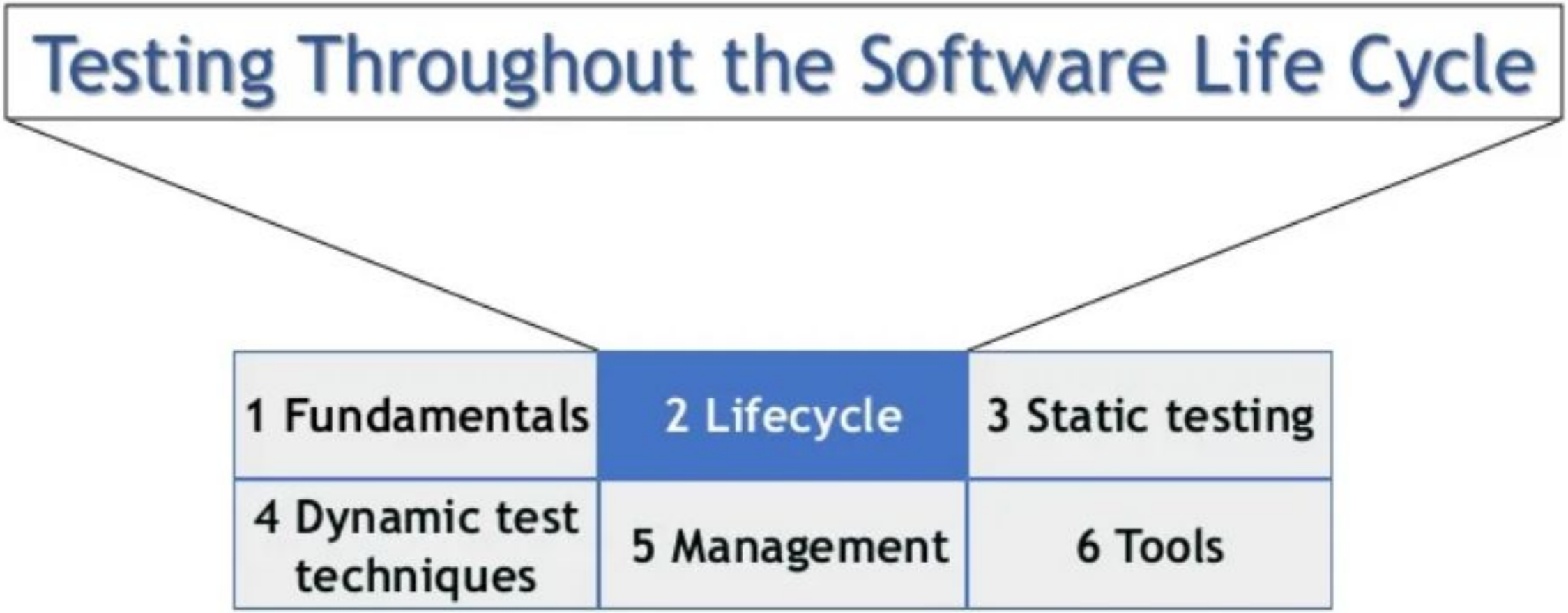
Benefits of s/w testing are enormous

- Quality
- Satisfied Customer
- Bringing Profit
- User Experience
- Business Optimisation

Skills Required by Tester

- Understanding Business needs
- Good communication skills
- Concepts of Testing
- Levels of Testing
- Knowing what to prioritize
- Continuous Learning
- Techniques for validation and verification
- Risk assessment and management
- Developing Test plan
- Execution of test plan

Testing throughout the software life cycle,

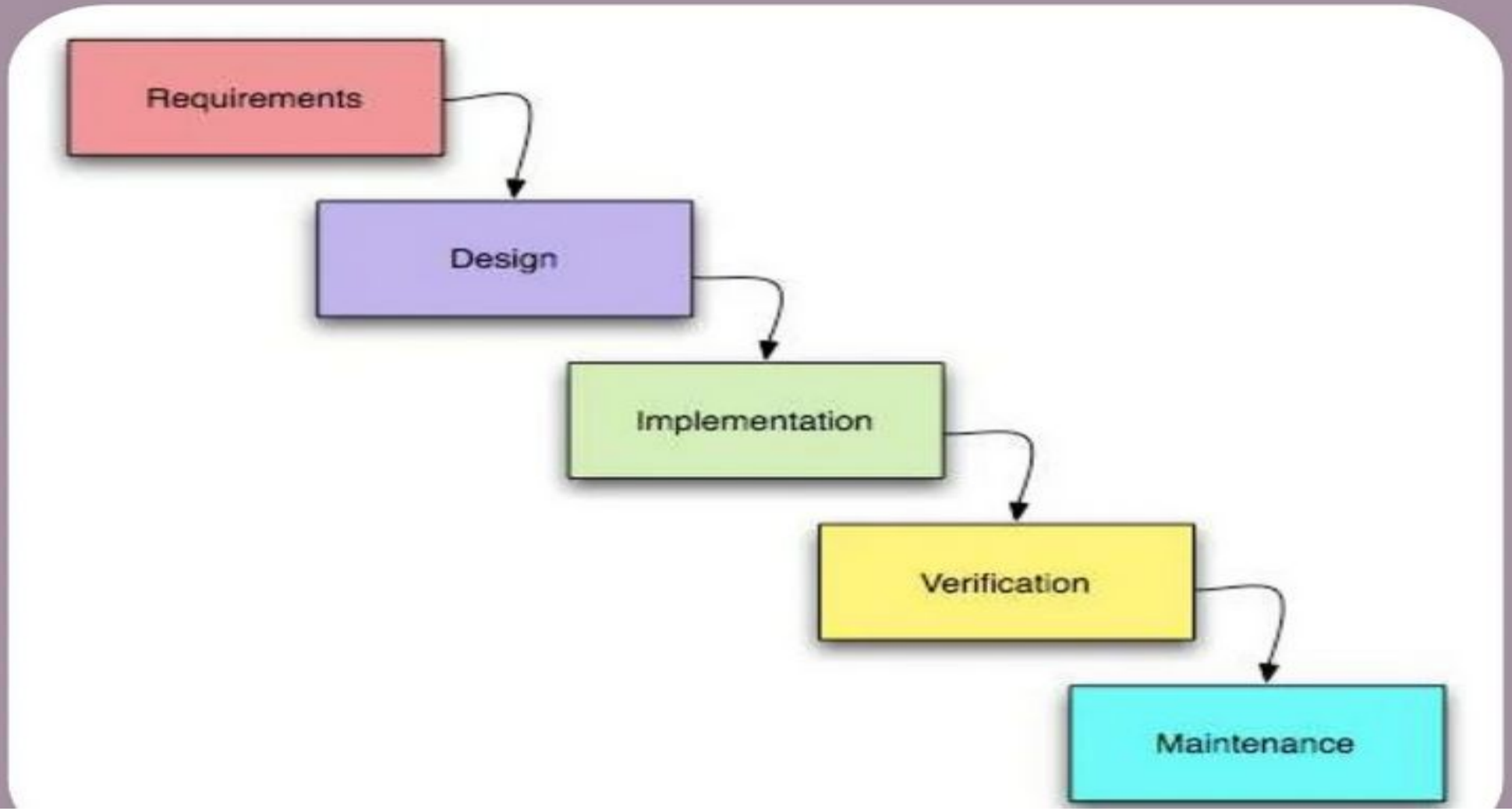


Software Development models

There are many software development models and methods are available in the market. All of this models are mentioned below:

1. Waterfall model
2. V model
3. Incremental model
4. RAD model
5. Agile model
6. Iterative model
7. Spiral model

Waterfall Model



Waterfall Model

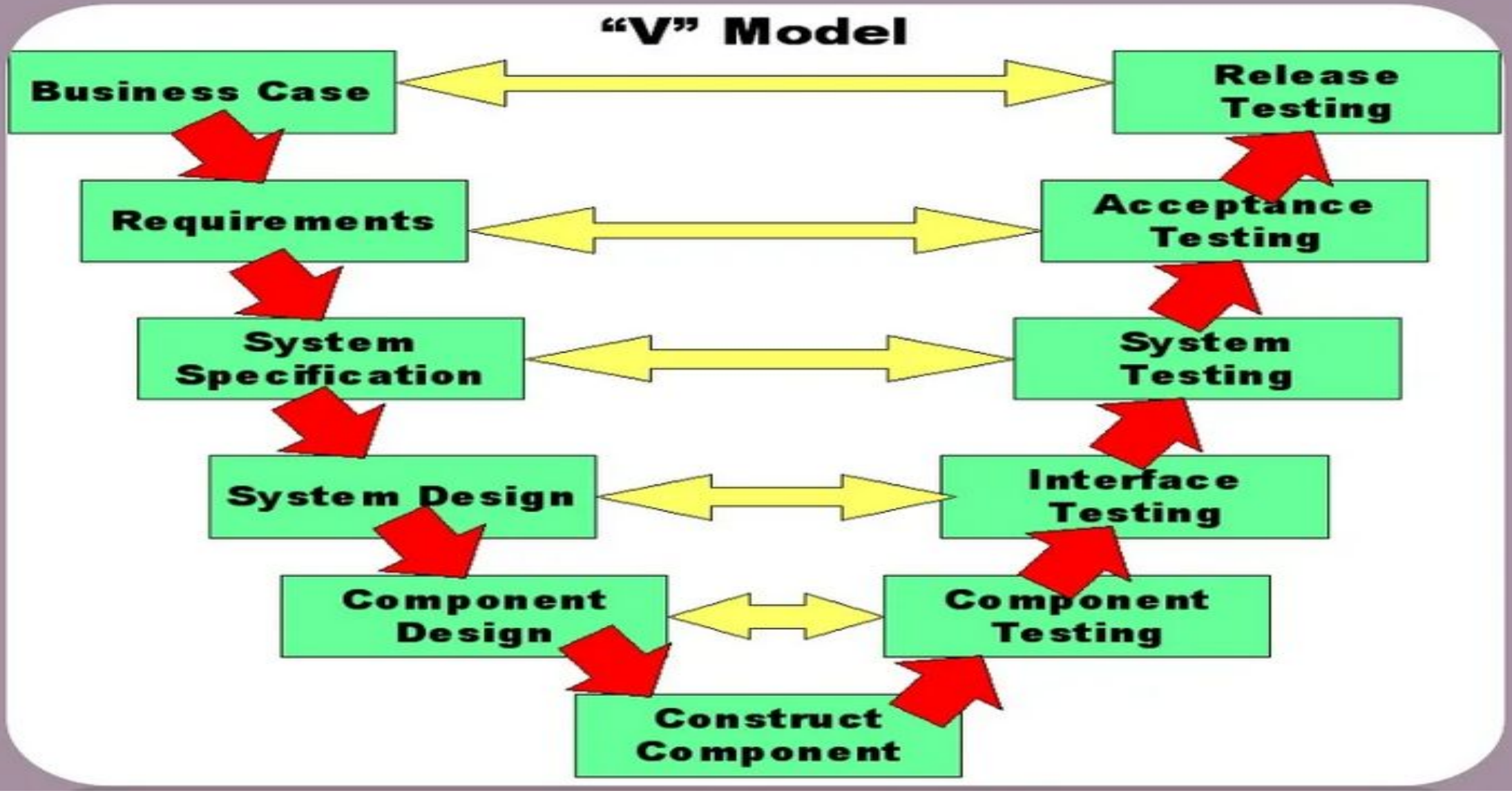
Advantages of Waterfall Model

- ✓ Simple and easy to understand and use.
- ✓ Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- ✓ Phases are processed and completed one at a time.
- ✓ Works well for smaller projects where requirements are very well understood.

Disadvantages of Waterfall Model

- ✓ Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- ✓ No working software is produced until late during the life cycle.
- ✓ High amounts of risk and uncertainty.
- ✓ Not a good model for complex and object-oriented projects.
- ✓ Poor model for long and ongoing projects.
- ✓ Not suitable for the projects where requirements are at a moderate to high risk of changing.

V Model


"V" Model
Business Case
Requirements
System Specification
System Design
Component Design
Construct Component
Component Testing
Interface Testing
System Testing
Acceptance Testing
Release Testing

V Model

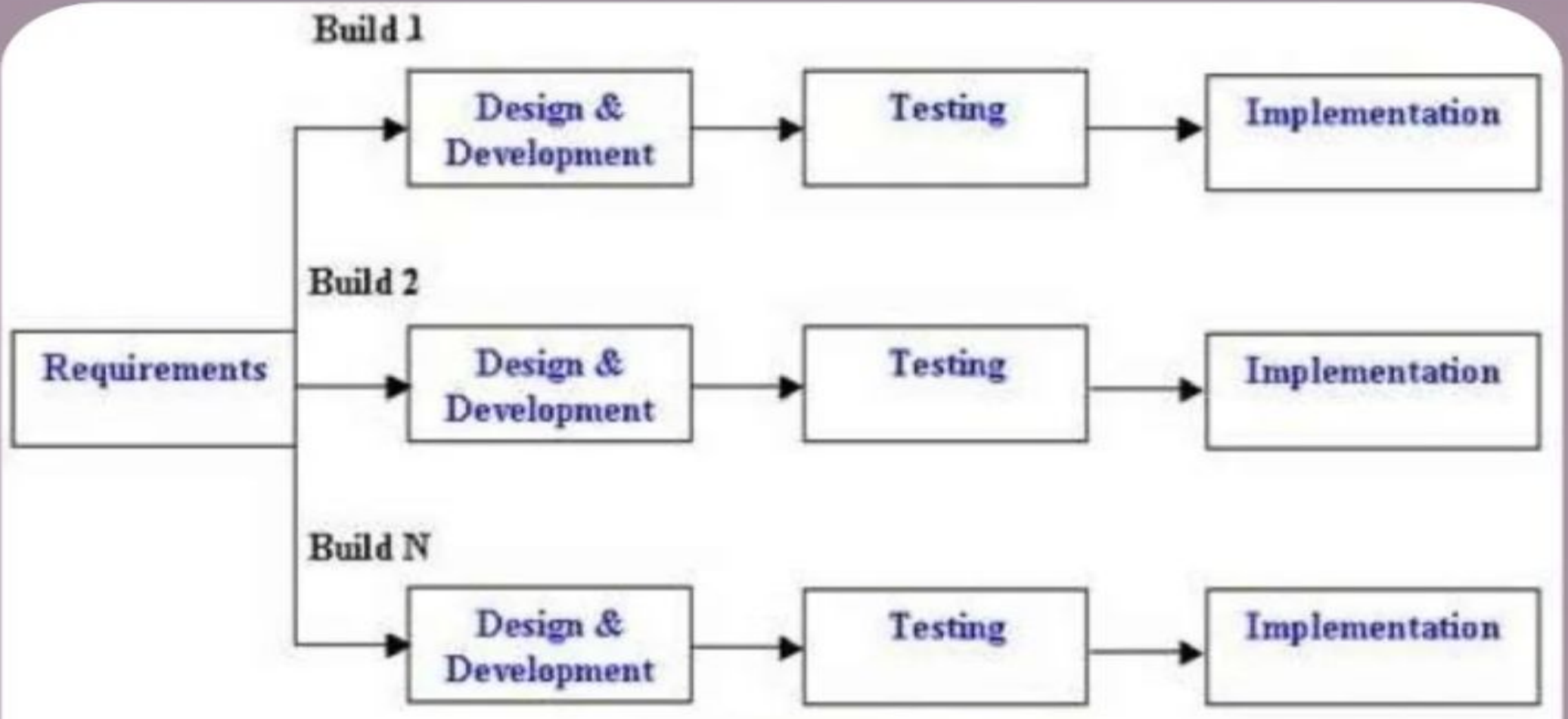
Advantages of V Model

- ✓ Simple and easy to use.
- ✓ Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
- ✓ Proactive defect tracking – that is defects are found at an early stage.
- ✓ Avoids the downward flow of the defects.
- ✓ Works well for small projects where requirements are easily understood.

Disadvantages of V Model

- ✓ Very rigid and less flexible.
- ✓ Software is developed during the implementation phase, so no early prototypes of the software are produced.
- ✓ If any changes happen in midway, then the test documents along with required documents has to be updated.

Incremental Model



Incremental Life Cycle Model

Incremental Model

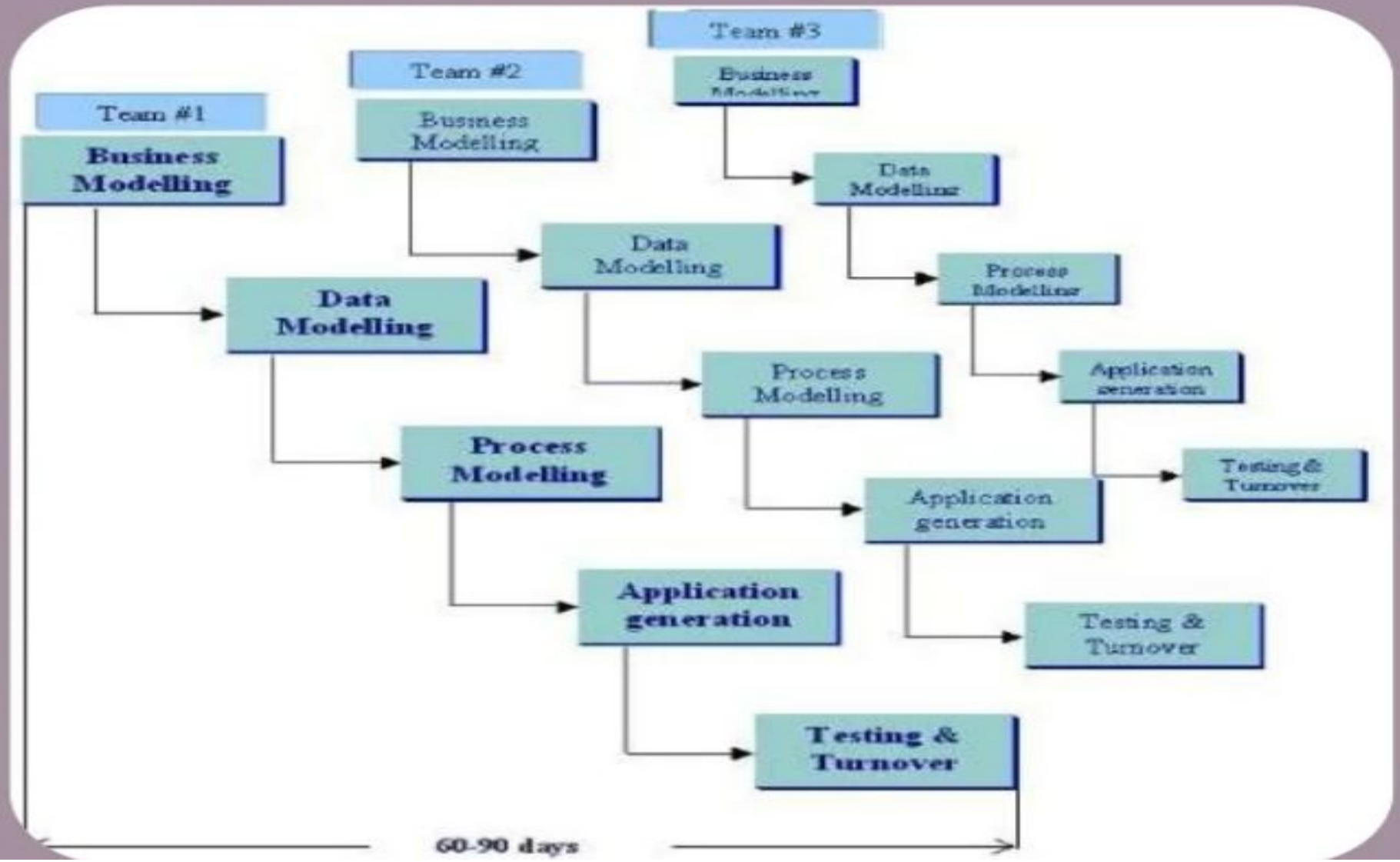
Advantages of Incremental Model

- ✓ Generates working software quickly and early during the software life cycle.
- ✓ More flexible – less costly to change scope and requirements.
- ✓ Easier to test and debug during a smaller iteration.
- ✓ The customer can respond to each bolt.
- ✓ Lowers initial delivery cost.
- ✓ Easier to manage risk because risky pieces are identified and handled during it'd iteration.

Disadvantages of Incremental Model

- ✓ Needs good planning and design.
- ✓ Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- ✓ Total cost is higher than a waterfall.

RAD Model


Team #1
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
Team #2
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
Team #3
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
60-90 days

RAD Model

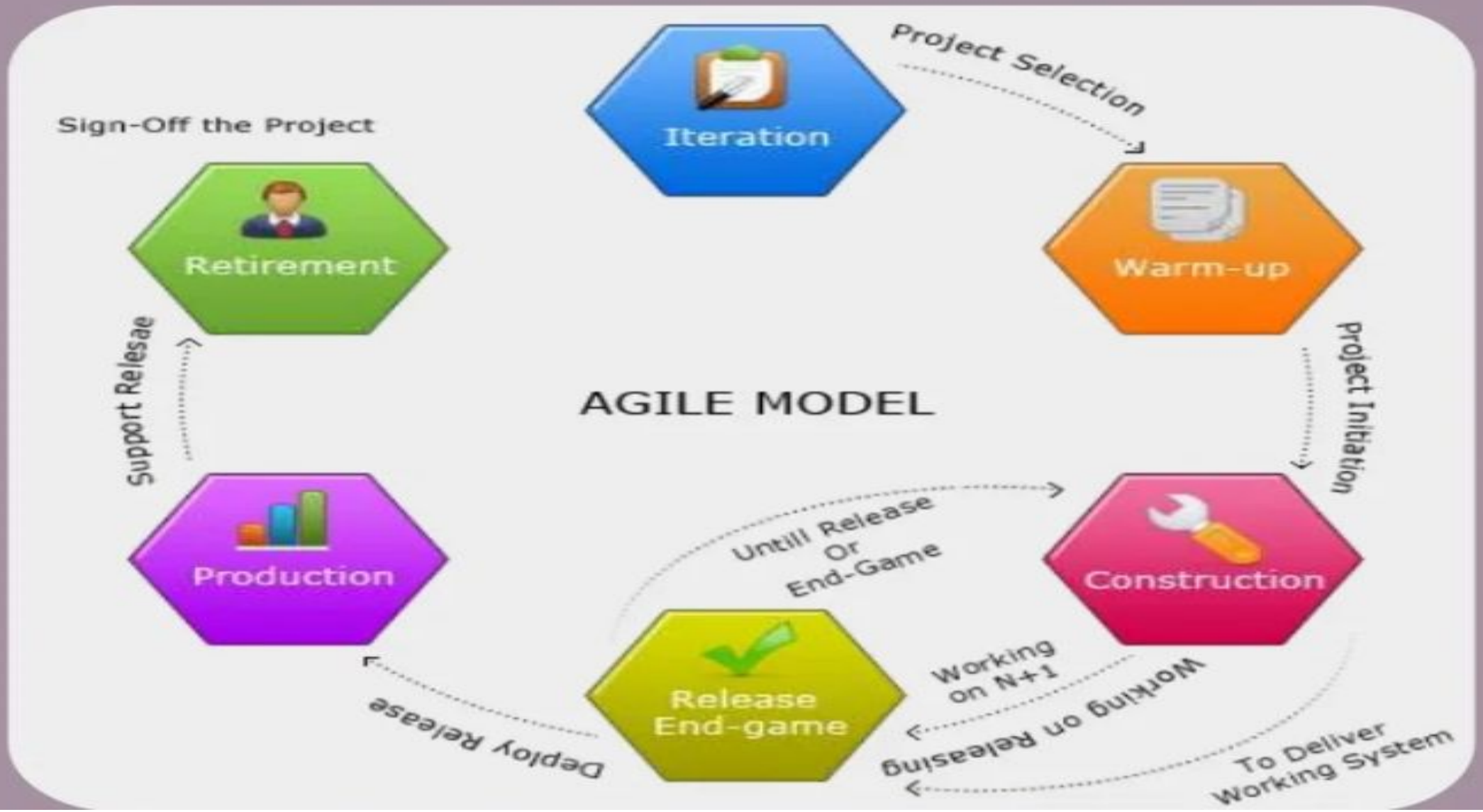
Advantages of RAD Model

- ✓ Reduced development time.
- ✓ Increases reusability of components
- ✓ Quick initial reviews occur
- ✓ Encourages customer feedback
- ✓ Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD Model

- ✓ Depends on strong team and individual performances for identifying business requirements.
- ✓ Only systems that can be modularized can be built using RAD
- ✓ Requires highly skilled developers/designers.
- ✓ High dependency on modeling skills

Agile Model


Iteration
Project Selection
Warm-up
Project Initiation
Construction
Untill Release
Or
End-Game
Release
End-game
Working
on N+1
Working on Releasing
To Deliver
Working System
Deploy Release
Production
Support Relesae
Retirement
Sign-Off the Project
AGILE MODEL

Agile Model

Advantages of Agile Model

- ✓ Customer satisfaction by rapid, continuous delivery of useful software.
- ✓ People and interactions are emphasized rather than process and tools.

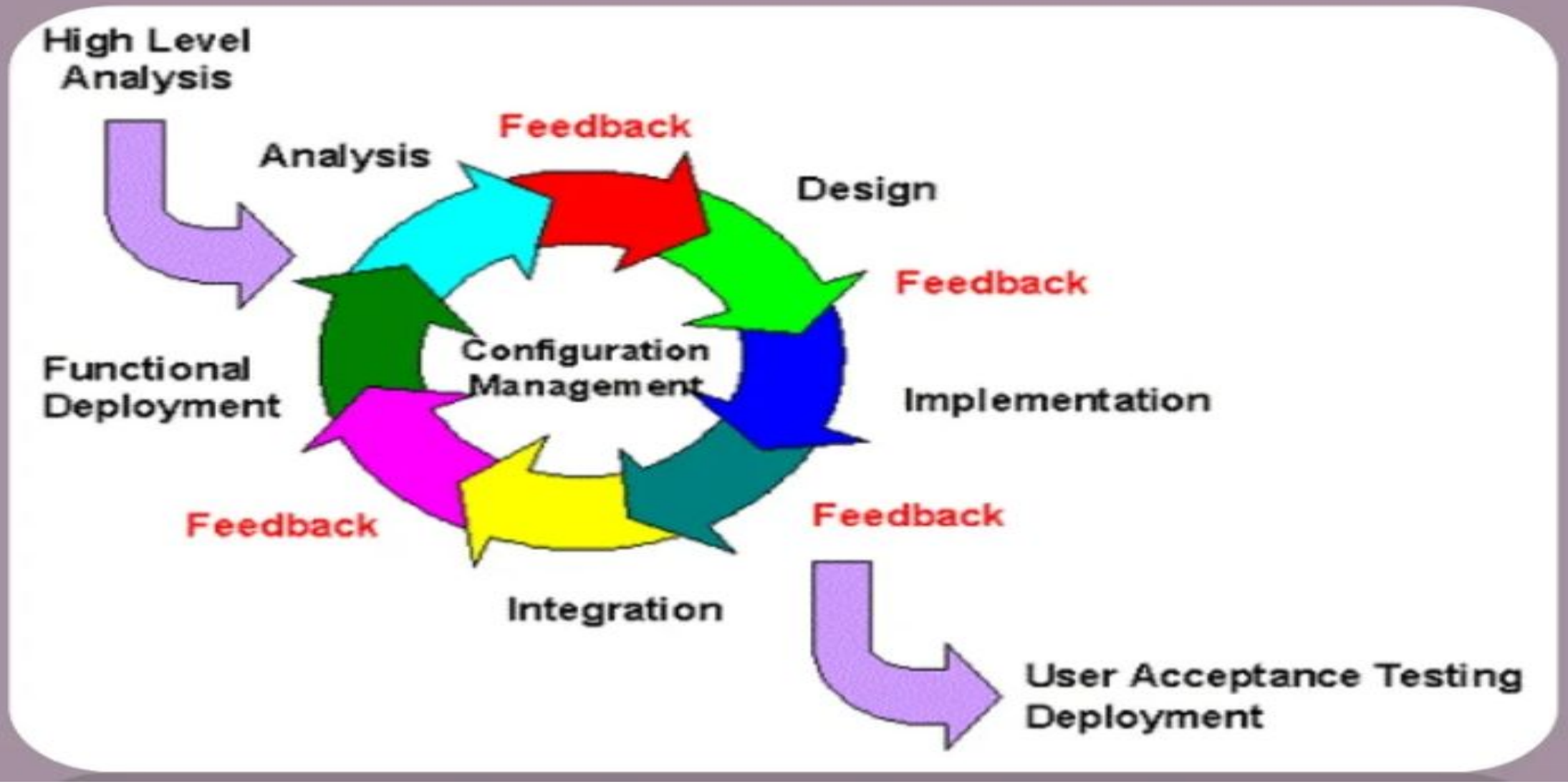
Working software is delivered frequently (weeks rather than months).

- ✓ Close, daily cooperation between business people and developers.
- ✓ Continuous attention to technical excellence and good design.
- ✓ Regular adaptation to changing circumstances.
- ✓ Even late changes in requirements are welcomed

Disadvantages of Agile Model

- ✓ In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- ✓ There is a lack of emphasis on necessary designing and documentation.
- ✓ The project can easily get taken off track if the customer representative is not clear what the final outcome that they want.

Iterative Model



Iterative Model

Advantages of Iterative Model

✓ In the iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.

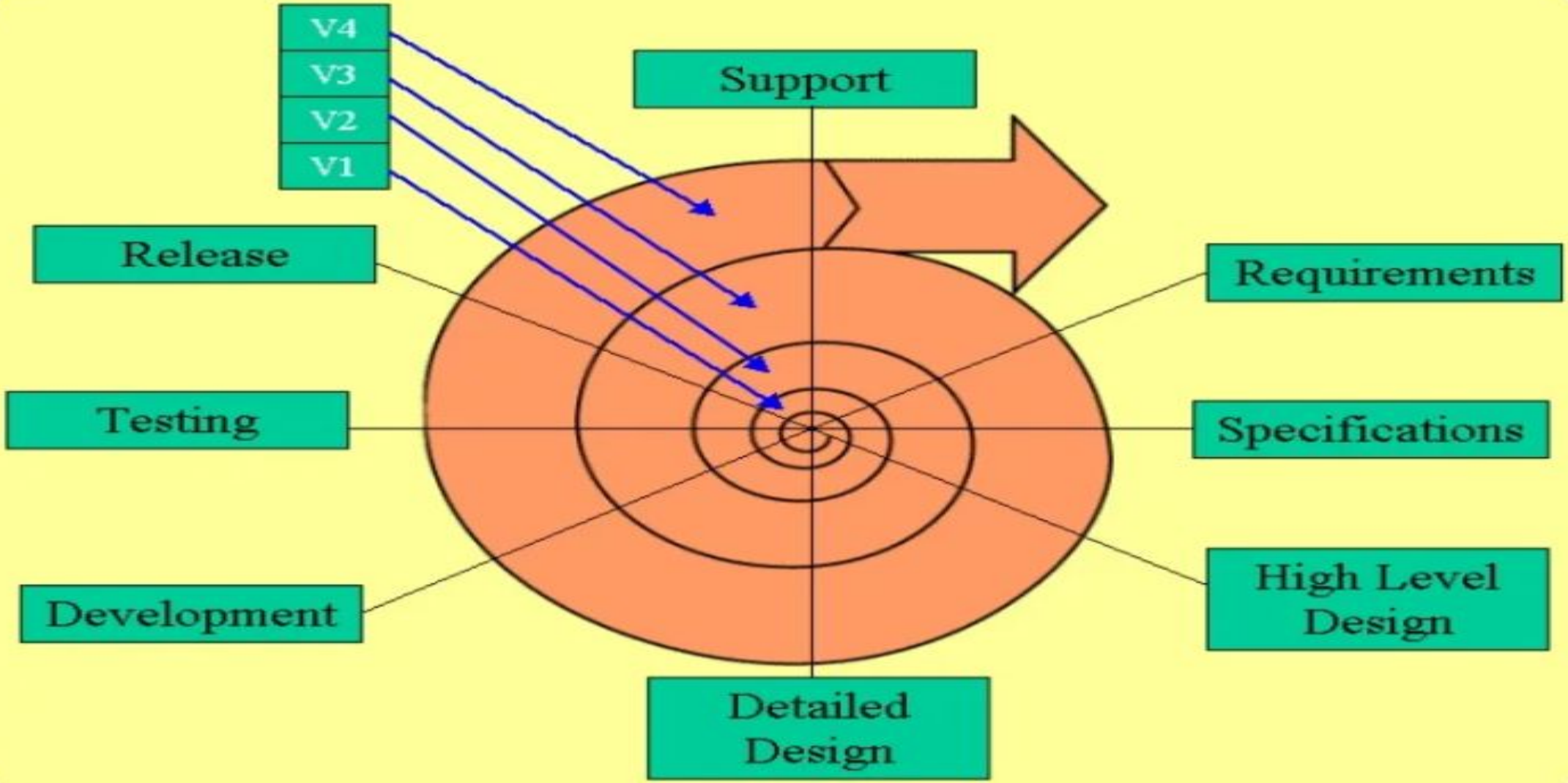
✓In the iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.

Disadvantages of Iterative Model

✓ Each phase of an iteration is rigid with no overlaps

✓ Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire Lifecycle.

Spiral Model



Spiral Model

Advantages of Spiral Model

- ✓ The high amount of risk analysis, hence avoidance of Risk is enhanced.
- ✓ Good for large and mission-critical projects.
- ✓ Strong approval and documentation control.
- ✓ Additional Functionality can be added at a later date.
- ✓ Software is produced early in the software life cycle.

Disadvantages of Spiral Model

- ✓ Can be a costly model to use.
- ✓ Risk analysis requires highly specific expertise.
- ✓ Project's success is highly dependent on the risk analysis phase.
- ✓ Doesn't work well for smaller projects.

Test levels

- Test Levels are to identify missing areas , prevent overlap and repetition between SDLC.
- Every phase of SDLC goes through testing .

Testing Levels

- Phases of software testing:
  - Unit Testing
  - Integration/Build Testing
  - Validation/Functional Testing
  - System Testing
  - Acceptance Testing

Unit Testing

- Test each module individually.
- Follows a white box testing (Logic of the program)

Integration testing

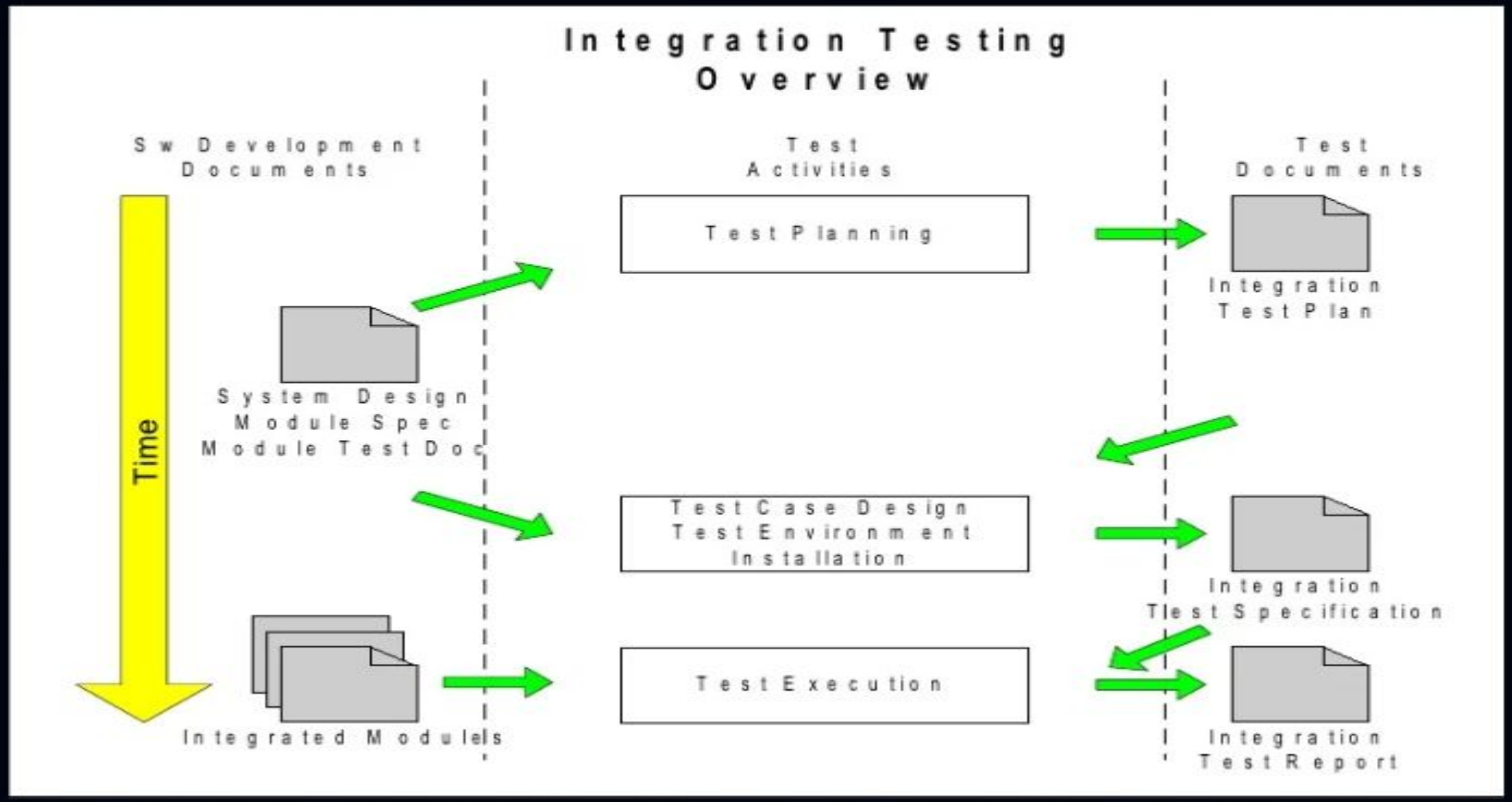
- Integrate two or more module.i.e. communicate between the modules.
- Follow a white box testing (Testing the code)

Integration Testing

- 1. Big Bang Integration Testing

a. Top Down
b. Bottom Up
c. Functional Increment
d. Component Integration Testing
e. System Integration

Integration Testing - Overview



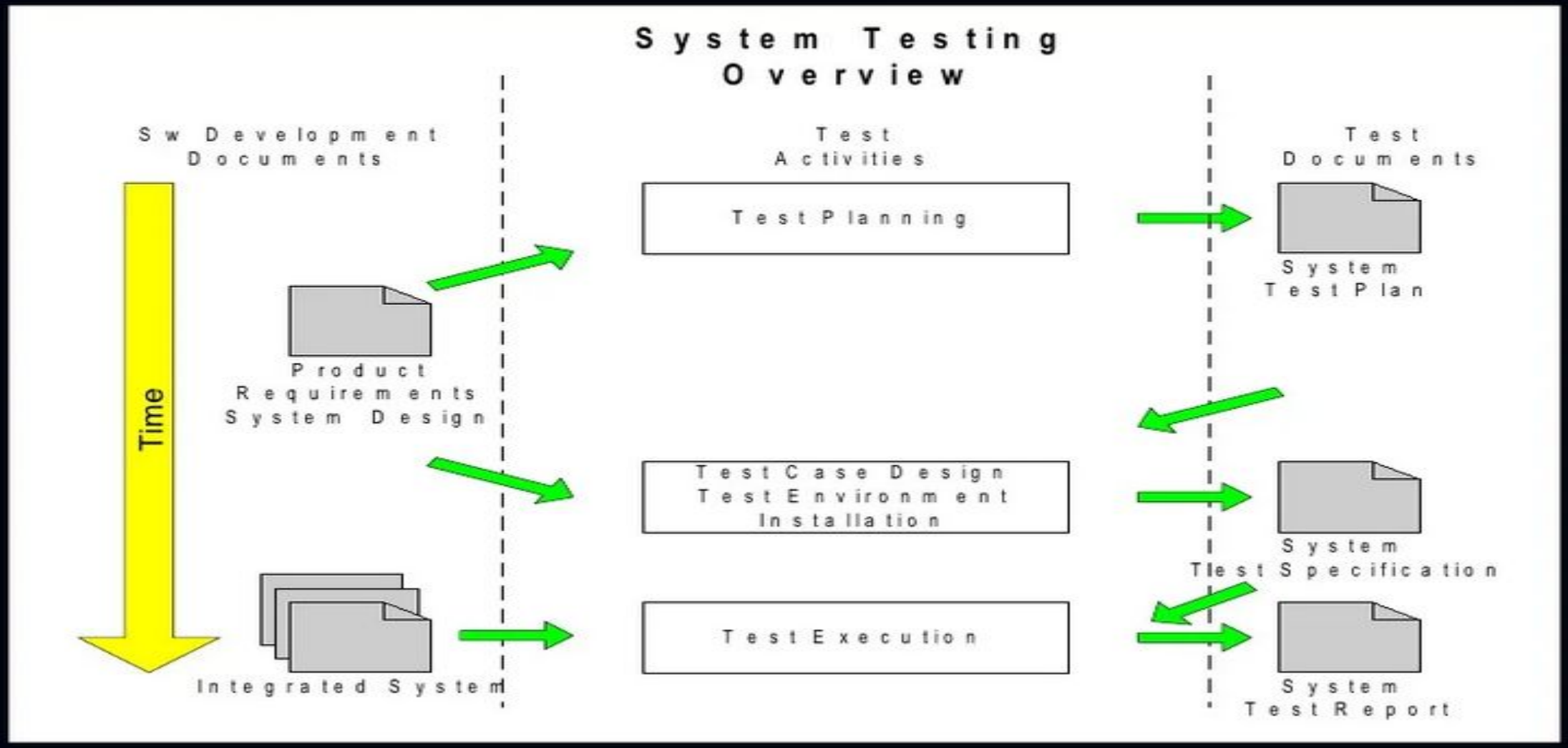
System Testing

- Confirms that the system as a whole delivers the functionality originally required.
- Follows the black box testing.

System Testing - Goals

- Incorrect or missing feature
- Performance errors
- Security errors
- User interface errors
- Configuration and compatibility
- Compliance to required standards

System Testing - Overview



System Testing - Overview

- System Tests:
  - Define test procedures and instructions
  - Review test plans
  - Execute test plans
  - Record results

System Testing - Strategy

- Developing System Tests
  - stress tests
  - security tests
  - recovery tests
  - performance tests

Acceptance Testing

1. Alpha Testing
2. Beta Testing

User Acceptance Testing (UAT)

- Building the confidence of the client and users is the role of the acceptance test phase.
- It is depend on the business scenario.
- Those functions required by the customer to demonstrate sufficient functionality and reliability to warrant acceptance.

V-Model

Test types : the targets of testing

- Functional Testing
  - Requirement Based Testing
  - Business Process Based testing
- Non Functional Testing
  - Functionality
  - Reliability
  - Efficiency
  - Maintainability
  - Portability
- Structural testing
- Change Related Testing (Confirmation or retesting and Regression testing)

Maintenance testing

- System is deployed for years
- Maintenance begins with the receipt of an application for a change or a release plan

1. On receipt of the new or changed specifications , corresponding test cases are specified or adapted
2. On receipt of the test object , the new and modified tests and the regression tests are executed
3. On completion of the testing , the testware is once again preserved .

- Triggers for Maintenance Testing

1. Modifications
2. Migration
3. Retirement

Software Maintenance

- Your system is developed…
  - It is deployed to customers…
    - What next?

- Maintenance
  - Categories of maintenance tasks
  - Major causes of problems
  - Reverse engineering
  - Management of maintenance activities

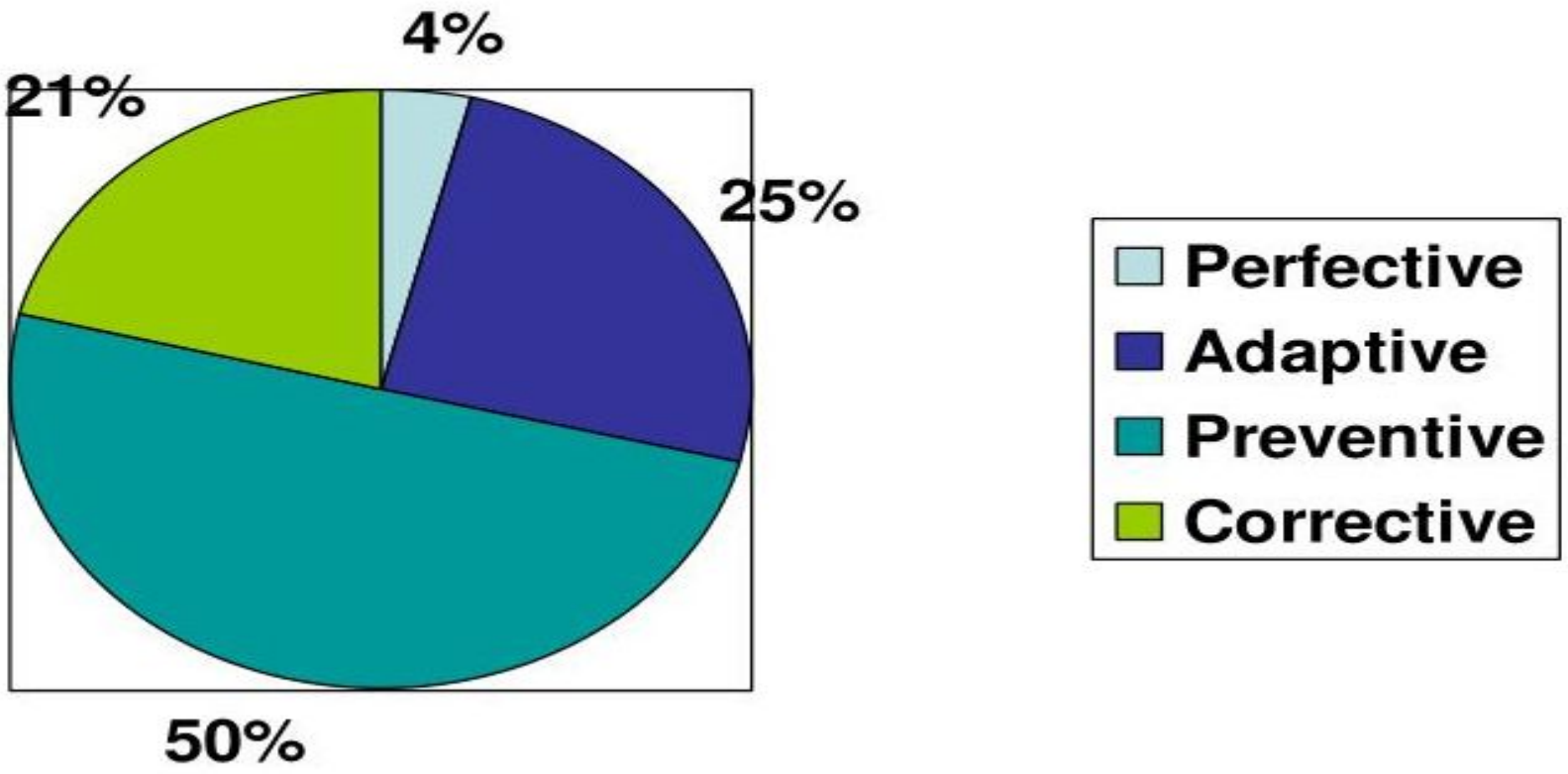
IEEE Definition

- Maintenance is
  - The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment.
  - More than fixing bugs!
- Estimates:
  - More than 100 billion lines of code in production in the world
  - As much as 80% is unstructured, patched, and badly documented

Maintenance Activities

- Corrective
  - Repair of faults that are discovered
- Adaptive
  - Modify software to changes in the environment, e.g. new hardware, new OS.
  - No change in functionality.
- Perfective
  - Accommodate new or changed user requirements.
  - Changes in functionality.
- Preventive
  - Activities aimed at increasing maintainability, e.g. documentation, code arrangement.

% Effort



- Maintenance is also 50% of the total lifecycle costs

Addressing Maintenance

- Higher quality code, better test procedures, better documentation, adherence to standards and conventions
- Design for change
- Prototyping and fine-tuning to user needs can help reduce perfective maintenance
- Write less code
  - Reuse

Laws of Software Evolution

- The laws of software evolution also force maintenance to occur
- Law of increasing complexity
  - A program that is changed becomes less and less structured and thus becomes more complex. One has to invest extra effort in order to avoid the increase in entropy.
- Law of continuing change
  - A system that is being used undergoes continuing change, until it is judged more cost-effective to restructure the system or replace it by a completely new version

Scant Knowledge Available

- Maintenance programmers generally lack detailed knowledge about the system or application domain
  - Problem in general, but worse for maintenance
  - Often scarce sources available to reference
  - Usually requires going to the source code to figure out how the system works
  - Experienced programmers have learned to distrust documentation, usually insufficient and out of date
    - When a quick-fix is done, the documentation is often not updated to reflect changes

Brief Summary of Unit 2

- **Fundamentals of Software Testing**
- **Historical perspective of testing**

1. Debugging oriented testing
2. Demonstration oriented testing
3. Destruction oriented testing
4. Evaluation oriented testing
5. Prevention oriented testing

- **Why testing is necessary**

1. Understanding of customer requirements may differ from person to person
2. Development people assume that whatever they have developed is as per customer requirements
3. Different entities are involved in different phases of software development
4. Integration issues can be present
5. There is a possibility of blindfold

- **TQM in cost perspective**

1. Green money / cost of prevention
2. Blue money / cost of appraisal
3. Red money / cost of failure

- **Characteristics of Big bang approach**

1. Testing is a last phase of development life cycle when everything is finalized
2. Characterized by huge rework, retesting, scrap, sorting
3. Regression testing reveals many issues as correction may not be correct
4. All requirements and designs cannot be covered in testing
5. Major part of software build never gets tested

Traditional definition of testing

- Testing is done to establishing confidence that the program does what it is supposed to do
- Any activity aimed at evaluating an attribute or capability of a program or system with respect to user requirements
- Process of demonstrating that errors are not present
- Testing gives number of defects present which indirectly gives a measurement of software quality
- Testing is done to evaluate the program or system used for making software
- Testing is used to confirm that a program performs it's intended functions correctly

What is testing?

- An activity of identification of the differences between expected results and actual results
- Process of executing a program with the intention to find defects
- Detecting specification related errors and deviations of application working from specifications
- Establish confidence that a program does what it is supposed to do
- Any activity aimed at evaluating an attribute of a program or software
- Measurement of software quality in terms of coverage of testing in terms of requirements, functionality, features, number of defects found etc can give information about confidence level imparted to customer
- Process of evaluating processes used in software development
- Verifying that system satisfies it's specified requirements as defined and is fit for normal use
- Confirming that program performs it's intended functions correctly
- Process of operating a system or component under specified conditions, observing and recording the results of such processing and making evaluation of some aspect of system
- Process of analyzing a software item to detect the difference between existing and required conditions
-

- Manager's view of a software testing
- The product must be safe and reliable for use
- The product will work under normal and adverse conditions
- The product is meeting exactly what user wants
- The processes used for development and testing are capable of finding the defects

- Tester's view of software testing
- Purpose of testing is to discover defects in the product and the process related to development and testing
- Process of trying to discover every conceivable fault or weakness in a work product

- Customer's view of software testing
- Testing must be able to find all the possible defects in the software and related documentation
- Testing must give a confidence that software users are protected from any unreasonable failure
- Testing must ensure that any legal or regulatory requirements are complied during development

- Basic principles of testing
- Define the expected output or result
- Developers should not test their own programs
- Inspect the results of each test completely and carefully
- Include test cases for invalid or unexpected
- Test the program to see if it does what it is not supposed to do as well as what it is supposed to do
- Avoid disposable test cases unless the program itself is disposable
- Do not plan tests assuming that no errors will be found
- The probability of locating more errors in any one module is directly proportional to the number of errors already found in that module

- Successful testers
- Testers must give confidence about the coverage of requirements, functionalities etc as defined in test plan
- Tester must ensure that user risks are identified before deploying the software in production
- Tester must conduct SWOT analysis of the software and processes used to make it

- Requirement traceability matrix
- Typical requirement traceability matrix is as shown below

Requirements	High Level Design	Low level design	Code files / Stored Procedures / TBLs	Test scenario	Test cases	Test results

- Requirement traceability matrix
- Typical requirement traceability matrix is as shown below

Problems with requirement traceability matrix

- Number of requirements is huge
- There may be one to many, many to one and many to many relationship between various elements
- One needs to update the requirement traceability matrix whenever there is a change
- Developing teams may not understand the importance of it
- Customer may not find value in it and may not pay for it

Risk traceability

- Typical risk traceability matrix is as shown below

Risk	High Level Design / Control	Low level design / Control	Code files / Stored Procedures / TBLs	Test scenario	Test cases	Test results

Important features of testing process

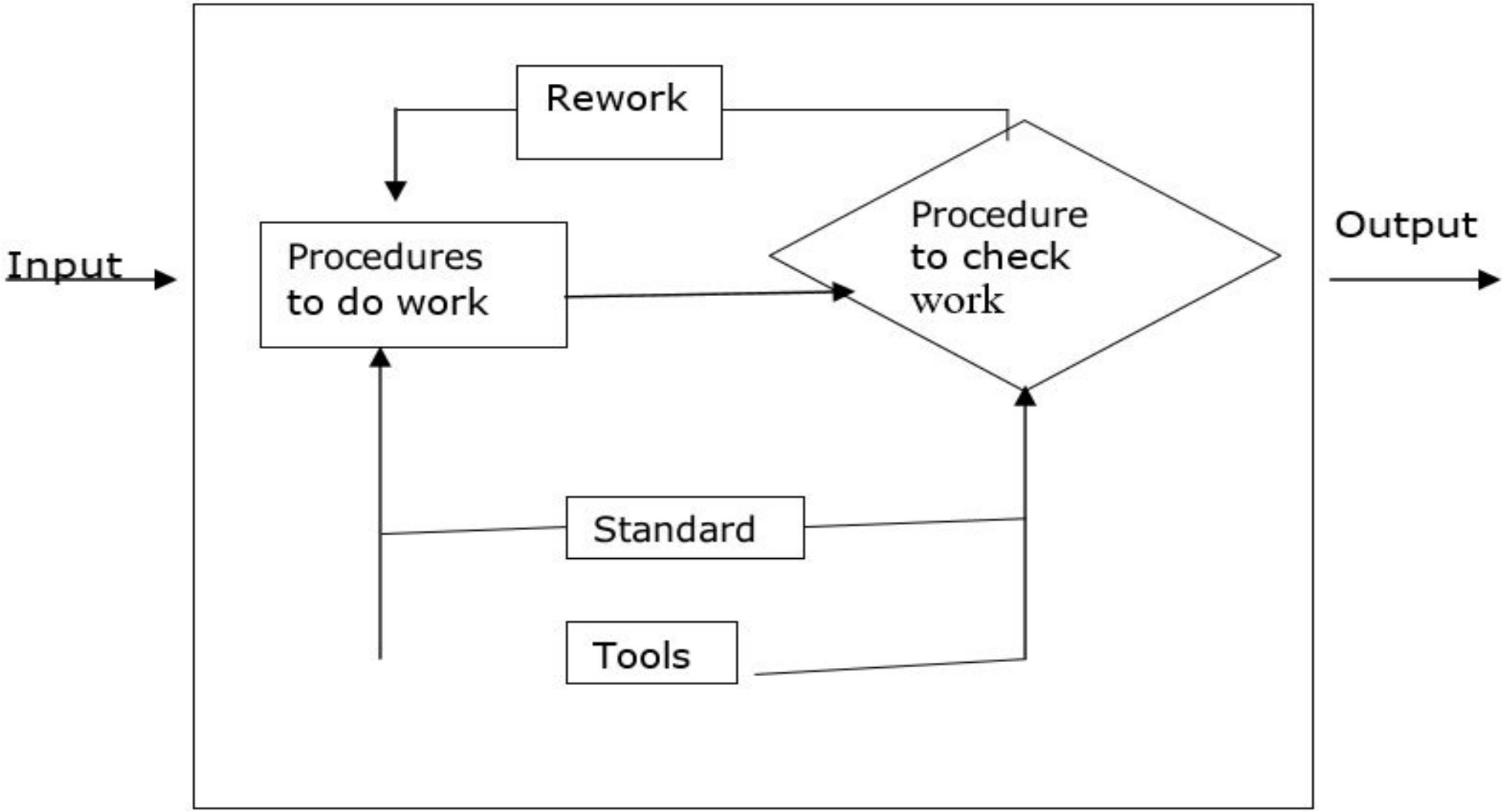
- Testing is a destructive process: Constructive destruction
- Testing needs a sadistic approach
- If the test doesn't detect a defect present, it is unsuccessful
- Test that detects a defect is a valuable investment, it helps in improving product
- It is risky to develop software and not to test it
- With a high pressure to deliver software as quickly as possible, test process must provide maximum value in shortest time
- Highest payback comes from detecting defect early in software development lifecycle and preventing defect leakage / defect migration
- Aim must be on defect prevention rather that finding and fixing a defect

Misconceptions about testing

- Defects found in testing are blames on developers
- Defects found by customer are blames on tester

- **Principles of software testing**
- Programmer / team must avoid testing their own work products
- Thoroughly inspect results of each test
- Initiate actions for correction, corrective action and preventive actions

Tester's workbench



Test planning
Plan testing efforts with an assumption that defects are there
If defects are not found, it is failure of testing activity
Successful Tester is not one who appreciates development but one who finds defect
Testing is not a formality to be completed at the end of development cycle

Differences between verification and validation

Verification	Validation
Verification is an activity where we check the work products with reference to standards, guidelines, procedures	Validation is an activity to find whether the software achieves whatever is defined by requirements
Verification is prevention based. It tries to check the process adherence.	Validation is detection based. It checks the product attributes.
Verification talks about a process, standard, guideline etc	Validation talks about the product
Verification is also termed as white box testing or static testing as the work product undergoes a review	Validation is also termed as black box testing or dynamic testing as workproduct is executed
Verification may be based on opinion of reviewer and may change from person to person	Validation is based on facts and is generally independent of a person

Verification can find about 60% of the defects	Validation can find about 30% of the defects
Verification involves • Reviews • Walkthroughs • Inspection • Audits	Validation involves all kinds of testing • System testing • User Interface testing • Stress testing etc
Verification can give • Statement coverage • Decision coverage • Path coverage	Validation can give • Requirement coverage • Feature coverage • Functionality coverage
Some parts of software can undergo verification only • Coding guidelines • Nesting • Commenting • Declaration of variables	Some characteristics of software can be proven by validation only • Stress testing • Performance testing • Volume testing • Security testing

Challenges in testing

- Requirements are not clear, complete, consistent, measurable and testable
- Requirements are wrongly documented and interpreted by business analyst and system analyst
- Code logic may be difficult to capture
- Error handling may be difficult to capture

- Other challenges in testing
- Badly written code introduces many defects
- Bad architecture of software cannot implement good requirement statement
- Testing is considered as a negative activity
- Testers find themselves in Lose - Lose situation

Location of test teams in organization Independent test team

- Advantages of independent test team
  - Test team is not under a delivery pressure
  - Test team is not under pressure of `Not finding' a defect
  - Independent view about a product is obtained as thought process of developers and testers may be completely different
  - Expert guidance and mentoring required by test team for doing effective testing may be available in form of test manager

- Disadvantages of independent test team
  - There is always `Us' vs `Them' mentality
  - Testers may not get a good understanding of development process
  - Sometimes management is inclined excessively towards development team or test team and other team feels that they have no value in an organization

- Test team reporting to development manager
- Advantages of test team reporting to development manager
  - There is a better cooperation between development team and test team as both are part of same team
  - Test team can be involved in development and verification / validation activities from the start of the project
  - Testers may get a good understanding of development process and can help in process improvement
-
- Disadvantage of test team reporting to development manager
  - Expert advice in form of test manager may not be available to testers
  - Sometimes development managers are more inclined towards development team
  - Many times testers start perceiving product from developers angle and their defect finding ability is reduced

Matrix organization

- Developers becoming testers Advantages of this approach
  - Developers do not need another knowledge transfer while working as a tester
  - Developers have better understanding of detail design, coding etc and can test application it easily
  - For automation, some amount of development skill is required in writing the automation scripts
  - It is less costly as there is no separate test team
  - Psychological acceptance of defects is not a major issue as developers themselves find the defects
-
- Disadvantages of this approach
  - Developers may not find value in doing testing
  - There may be blindfolds while understanding requirements or selection of approach
  - Developers may concentrate more on development activities
  - Development needs more of a creation skill while testing needs more of a destruction skill

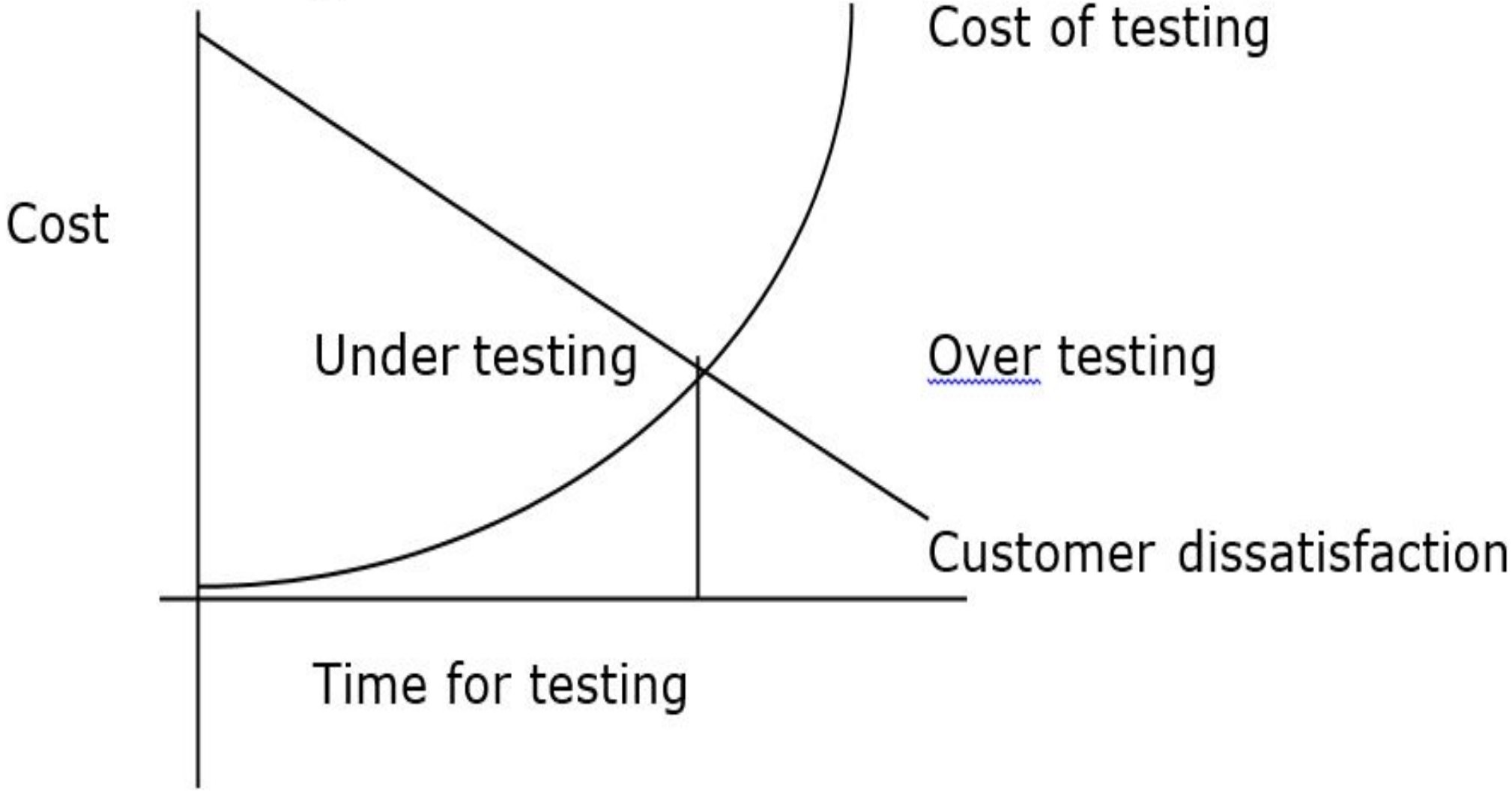
Independent testing team

- Advantages of this approach
  - Separate test team is supposed to concentrate more on test planning, test strategies and approach, creating test artifacts etc
  - There is independent view about the work products derived from requirement statement
  - Special skills required for doing special tests may be available in such independent teams
  - `Testers working for customer', can be seen in such environment
-
- Disadvantages of this approach
  - Separate team means additional cost for organization
  - Test team needs ramping up and knowledge transfersimilar to development team
  - Organization may have to check for rivalries between development team and test team

Domain experts doing a software testing

- Advantages of this approach
  - `Fitness for use' can be tested
  - Domain experts may provide facilitation
  - Domain experts understand the scenario faced by actual users
-
- Disadvantages of this approach
  - Domain experts may have prejudices about the domain
  - It may be very difficult to get domain experts in all areas
- It may mean huge cost for the organization

Economics of testing



Cost of testing curve is guided by

- Defect finding ability of testing team
- Defect fixing ability of development team
- Defect introduction index of development team

- Cost of customer dissatisfaction is guided by,
- Cost of customer dissatisfaction mainly depends upon customer supplier relationship
- Cost of customer dissatisfaction also depends on the type of the product and it's mission criticality

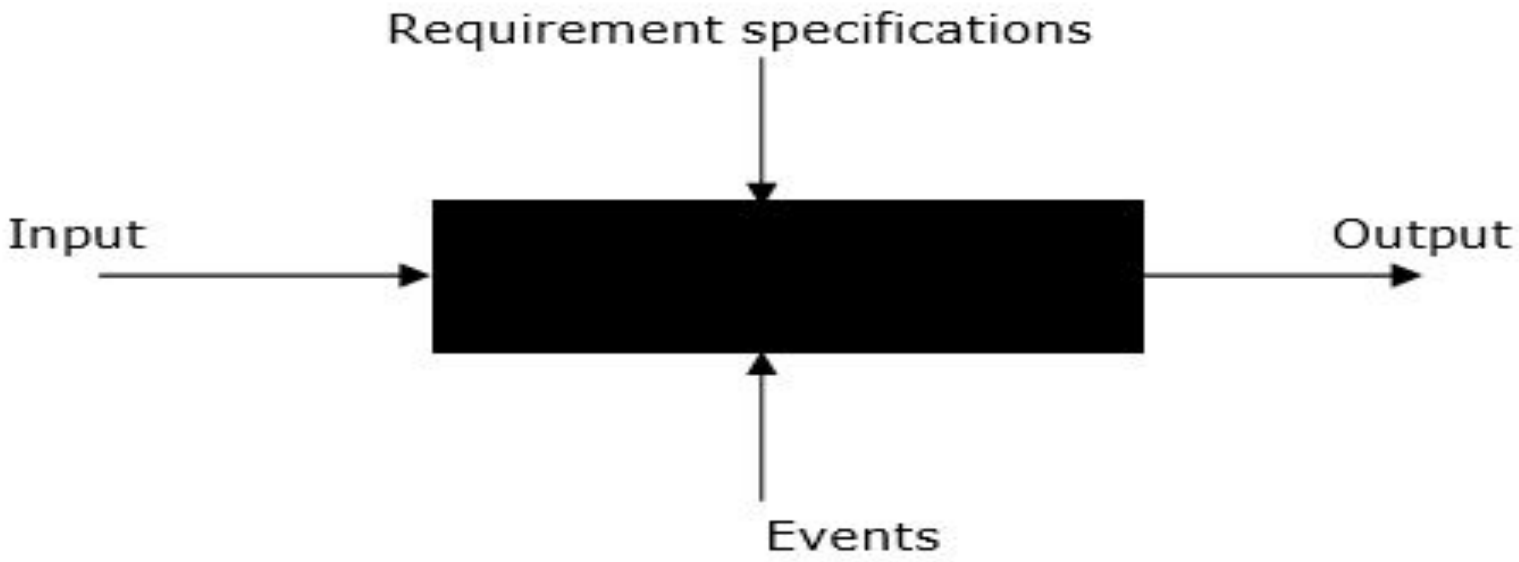
Testing process

- Defining test policy
- Defining test strategy
- Preparing test plan
- Establishing testing objectives
- Designing test scenario and test cases
- Writing / reviewing test cases
- Defining test data
- Creation of test bed
- Executing test cases
- Test result
- Test result analysis
- Performing retesting / regression testing
- Root cause analysis and corrective / preventive actions

Attitude towards testing

- New members of development team are not accustomed to view testing as a discovery process
- Developers may not accept the defect in first place
- Conflict between developer and tester can create differences between project teams and test teams
-

Black box testing (Domain testing / specification testing)



Advantages of `Black Box Testing'

- Black box testing is the only method to prove that software does what it is supposed to do
- It is the only method to show that software is living and it really works
- Some types of testing can be done only by black box testing methodologies

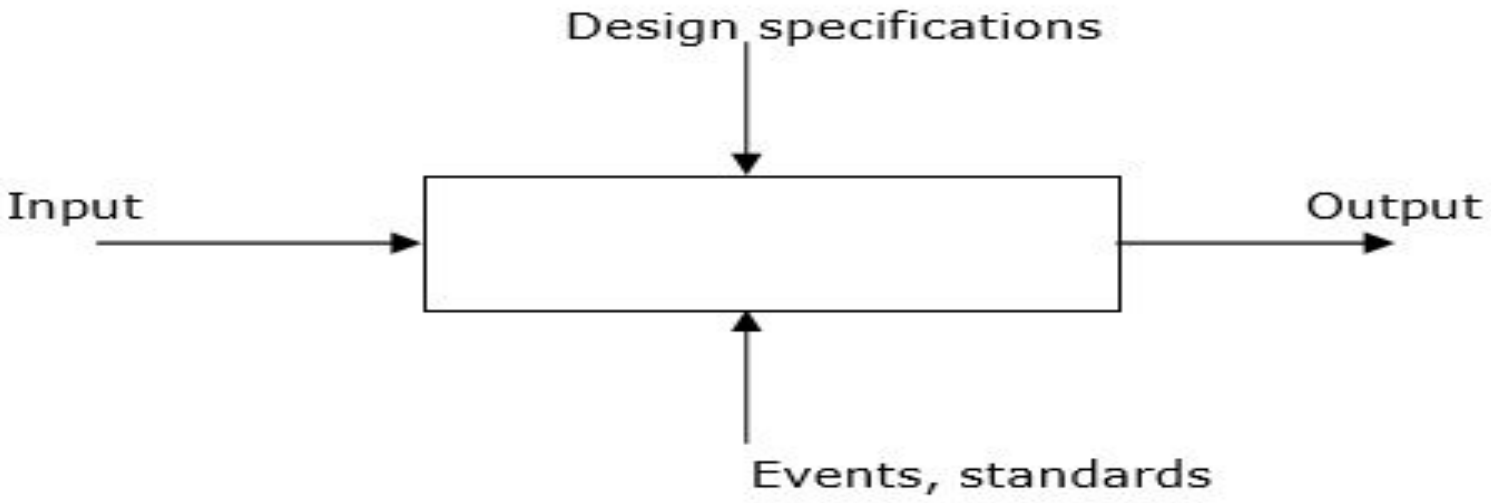
Disadvantage of `Black Box Testing'

- Some logical errors in coding can be missed
- Some redundant testing is possible as requirements may execute same branch of code again and again

Test data definition

- Equivalence partitioning
- Boundary value analysis
- Cause and effect graph
- State transition testing
- Use case based testing
- Error guessing

White box testing



Advantages of 'White Box Testing'

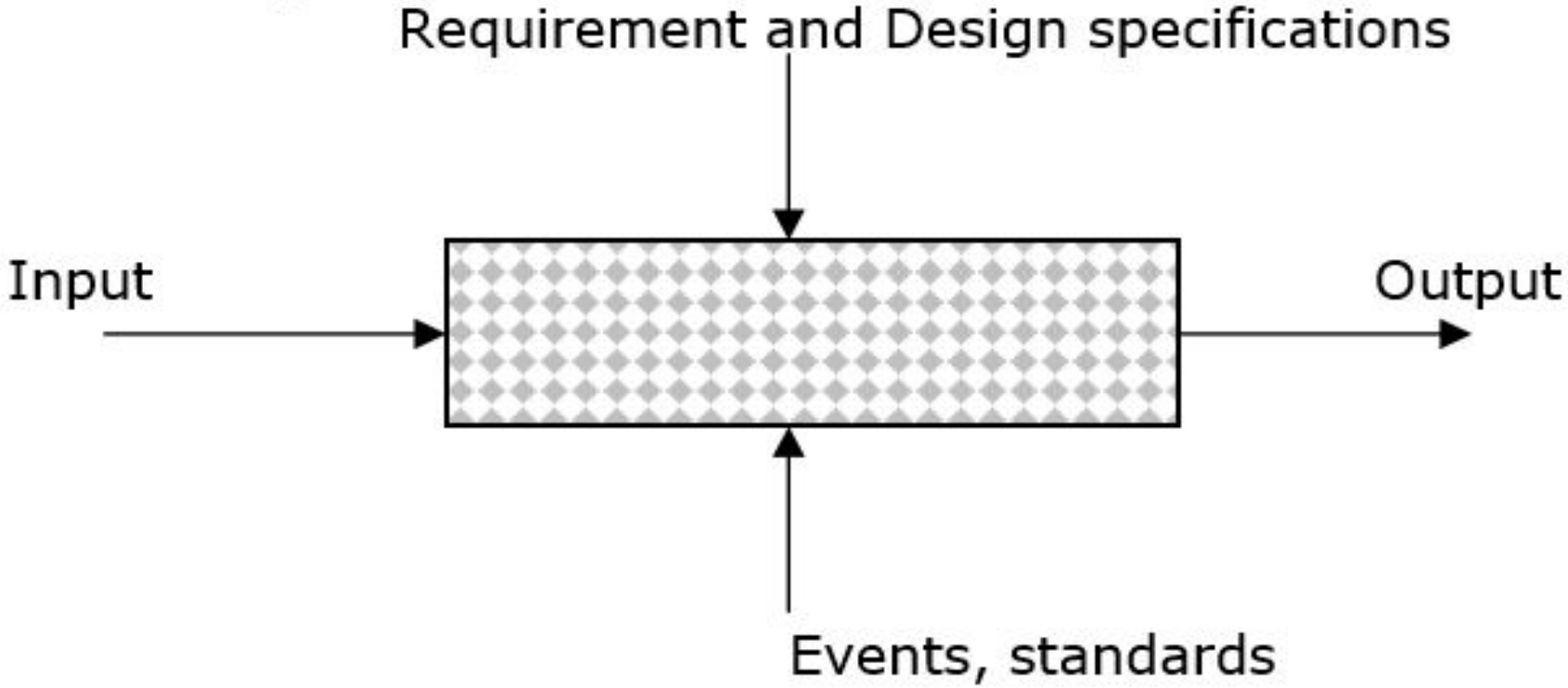
- Only white box testing can ensure that defined processes, procedures, methods etc of development are really followed
- White box testing or verification can give early warnings if something is not done properly

Disadvantage of `White Box Testing'

- Some characteristics of software work product can be verified only
- It does not ensure that user requirements are met correctly or not
- It does not establish whether decisions, conditions, paths, statements covered during reviews are sufficient or not
- Sometimes white box testing is dominated by the usage of checklists

- Test case designing
  - Statement coverage
  - Decision coverage
  - Condition coverage
  - Path coverage
  - Logic coverage

Gray box testing



Advantages of `Gray Box Testing'

- Combine the advantages of White box testing and Black box testing

Disadvantages of `Gray Box Testing'

- Generally Gray box testing is conducted with some automation tools

People challenges in software testing

- Tester is responsible for improving testing process to ensure better customer satisfaction and better products with less number of defects going to customer
- All the defects must be found and the confidence level must be built in the process that can give customer satisfaction
- Proper coverage as required by test plan must be achieved
- Testing needs trained and skilled people who can make it successful by delivering product with minimum defects to the stakeholders
- Testers have to improve their skills through continuous learning
- Tester needs a positive team attitude for creative destruction of software
- Defect in the software is not an opportunity to blame developers but improve the product
- Testers must be able to pin point the lacunae of software development process as the defects are found
- Testing is creative work and is a challenging task
- Feasible test scenario and test cases, effective ways of looking for the defects is essential to improve testing effectiveness
- Programmers and testers work together to improve quality of software developed and delivered to customer and the process used for software development and testing
- Testers hunt for defects, pursue defects not people including developers
- Testing needs patience, fairness, ambition, creditability, capability, diligence on part of testers

Tester's role

- Calculate testing cost, effectiveness of testing and ensure that management understands it
- By doing good testing, number of customer complaints must reduce and cost of failure must go down
- Demonstrate cost reduction, increase in effectiveness over a time span as the rework, scrap etc reduces over a longer horizon
- Highlight needs and benefits of training people in test team as well as development team on testing activities and skills so that testers can perform better
- Collect and distribute information on testing to all team members as well as development team / organization which can be used for improvement
- Get involved in test budgeting

Skills required by tester

- Written and oral presentation
- Effective listening
- Facilitation
- Software development, Operations and Maintenance
- Continuous education
- Concepts of testing
- Levels of testing
- Techniques for validations and verifications
- Selection and use of testing tools
- Knowledge of testing standards
- Risk assessment and management
- Developing test plan
- Defining acceptance criteria
- Checking processes
- Execution of test plan
- Continuous improvement of testing process

THANKS