Test Design Techniques

JAN, 2026

expleo

Contents

- Test Strategy and Plan.
- Test Case and Test Scenario.
- Categories of Testing Techniques.
- Black-box and White-box Techniques.
- Static Testing Techniques.
- Review Process.
- Quiz.

Test Strategy and Plan

What is Test Strategy?

- **Test strategy : describes the organization's general, project-independent methods for testing.**
- The objective of the Test Strategy is to provide a systematic approach to the software testing process in **order to ensure the quality, traceability, reliability and better planning.**
- And it also describes what kind of technique must be used and which module will be tested.
- It contains various components like **documentation formats, objectives, test processes, scope, customer communication strategy, etc.**
- The test strategy helps in **planning the testing phase**. It has a brief introduction about how testing needs to be done and what methods to be followed **depending upon the type of project/requirement.**

What is Test plan?

- The test plan is a document that describes the **scope, objective, method of software testing**.
- It is a **project plan for testing tasks** at hand.
- Documenting a test plan helps **to communicate** with other teams within **the project, the Manager**, and **other stakeholders**.
- Each software project team can have its own test plan based on the requirement.
- The test plan has a brief introduction about the methods and type of testing, which is going to be used during the testing.

Test plan Attributes

Test plan covers :

- Test Objectives
- In-Scope and out of scope items
- Risks
- Approach (extent of testing, automation/regression/performance test)
- Test Schedule

Test plan Attributes

Test plan covers :

- Entry and Exit Criteria
- Pass/Fail criteria
- Test Deliverables (Reports, test cases, etc)
- Environmental needs
- Staffing needs

Difference between Test Plan and Test Strategy

#	Test Plan	Test Strategy
1	A test plan is derived from Software Requirement Specification (SRS), describing in detail the scope of testing and the different activities performed in testing.	A test strategy is a high-level document describing the way testing is carried out.
2	A test plan is project level.	A test stategy organization level
3	It describes the whole testing activities in detail - the techniques used, schedule, resources etc.	It describes the high-level test design techniques to be used, environment specifications etc.
4	It is prepared by test lead or test manager.	It is generally prepared by the project manager.
5	Components:The major components of Test Plan include – Test Plan ID, test environment, features to be tested, entry/exit criteria, status, type of testing, brief introduction etc	The major components of Test Strategy include – Scope, Objective, business issues, risks, testing approach, testing deliverables, defect tracking, training, automation etc.
6	A Test Plan usually exists individually.	Test strategy is divided into multiple test plans that are taken care further independently.

Testcase and Test Scenario

Introduction and Objective of Test Design

- Test Design is a part of STLC Process, and this session details the activities performed as a part of Test Design Process.

Objective of Test Design:

User will understand the process of,

- Test Design Techniques
- Test Scenario
- Test Cases
- Test Steps
- Test Data
- RTM
- Test Environment

Test Scenario

- In Requirement Analysis phase, the **testable requirements are identified** and then transformed into **Test Scenarios during the test design phase.**
- Test Scenarios helps to **break down the requirement document into simple and common functionalities** are grouped together.
- The Testable Requirements are identified during the **Test Requirement Analysis** and then transformed into **Test Scenarios during the Test Development Phase**.
- Test Scenarios are identified once the **Test Requirements have been examined and high-level Test Scenarios have been identified.**

Advantages of Test Scenario

- During the **Requirement Analysis**, test scenarios are identified at a very high level and can act as a **tool to guarantee** that the **Requirements** are **understood completely and clearly**.
- Test Scenarios help in **ensuring** that **all testable requirements are covered**.
- The test scenarios are subsequently converted into test cases, verifying that all requirements are met.
- Test Scenarios are effective because they are written in the same order as they would be used in the actual world.
- They can also **detect problems** that could go **undetected during the execution of individual test cases**.

Types of Test Scenario

Test Scenario Types:

- Basic
- Alternate
- Exceptional

➢ **Basic :** Basic flow - Scenarios that assure the application achieves its primary goal or purpose.

➢ **Alternate** : Scenarios created to attain the aim through a different workflow.

➢ **Exceptional** : Scenarios created to test the application's behavior under unusual circumstances.

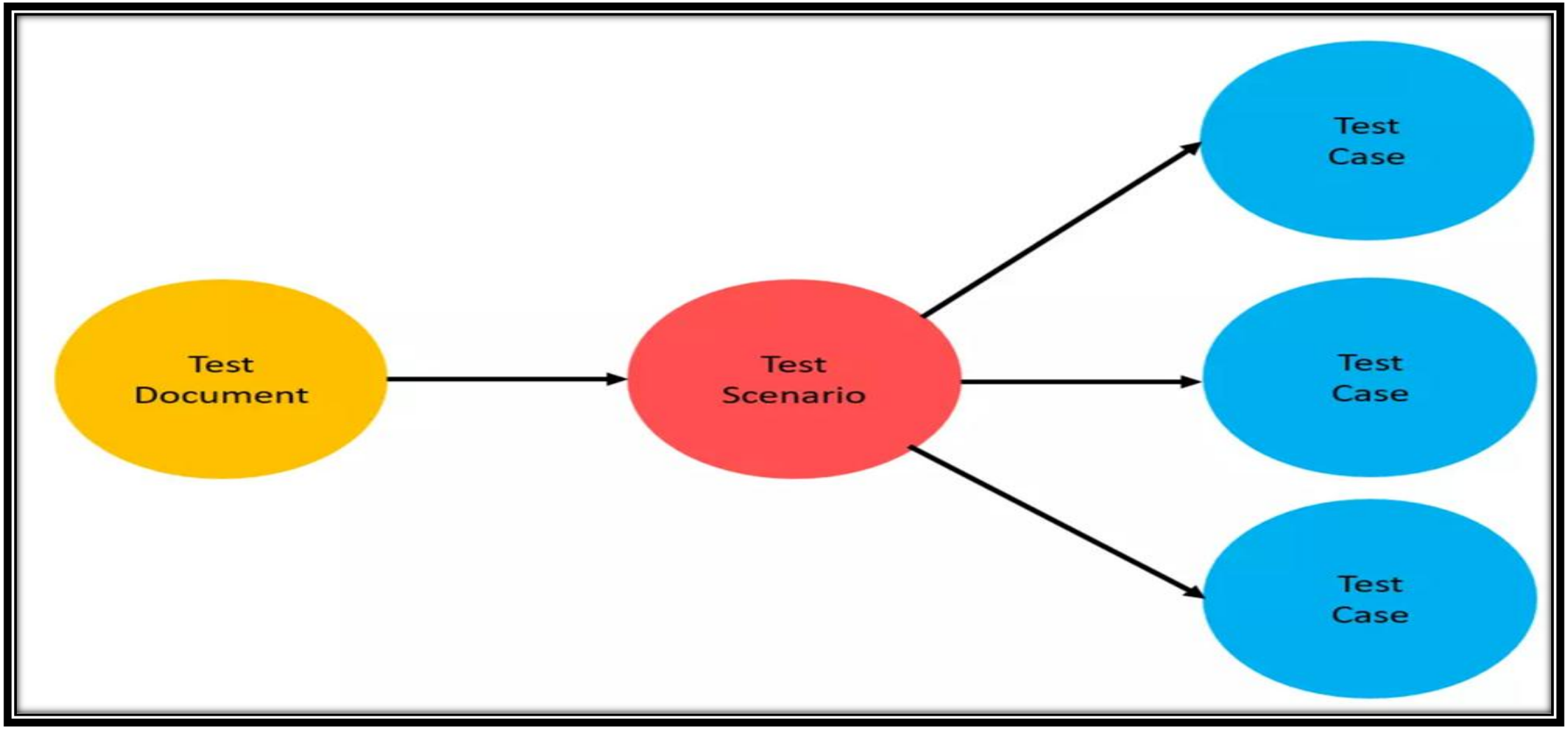
Components of Test Scenario:

- The components of a test scenario required to construct test scenarios for any application are listed below.
  - –Test Scenario ID
  - –Test Scenario Name
  - –Test Scenario Description
  - –Requirement ID

A sample template depicts the test scenario:

Test Scenario ID	Test Scenrio Name	Test Scenario Description	Requirement ID	Comments

Graphical representation of a test scenario :



Test Cases – An Overview

- A **combination of inputs**, **execution preconditions**, and **expected outputs** established for a **specific goal**, such as testing a programme route or verifying compliance with a requirement.
- A high likelihood of finding an error.
- Aids in the **discovery of information**.
- Must be written for the invalid and unexpected, as well as the valid and expected conditions.
- A Test case condition should not be redundant, and it should not be too simple or too complicated (more than one expected result).
- The purpose of the test or a description of what need is being tested should not be duplicated and neither too simple nor too complicated (more than one expected outcome).

Test Cases – An Overview

Importance of Test Case:

- Test cases are generally written by a **Test team** or otherwise called as a **QA Team**.
- A test case is described in simple words, along with test steps for every test case.
- Once a system feature or group of features has been completed by the development team, testing can commence.
- A **test suite is a collection or set of test cases**.
- Test steps, test data, preconditions, and postconditions that check requirements are all included in a test case document.

Test Cases – An Overview

Importance of Test Case:

- Test cases explain **what must be done to test a system**, such as the **actions** that must be performed.
- The input data to check the test condition for every test case and the expected results throughout the test case execution.
- **Developers and testers** can use test cases **to find faults that happened during development** or **defects that were missed during unit testing**.

Test Case Productivity Guidelines

- A Test Case written with the Test Steps in the active case will assist the tester in easily understanding the Test Case and successfully completing the testing.
- A good length for an individual Test Case is 10-15 steps.
- Reusing Test Cases during Test Case preparation will help to boost productivity (i.e., the number of Test Cases/Test Steps produced per day).
- The productivity of test cases can be increased by employing test management tool.

Test Case Template

Test Scenario ID	Test Case ID	TestCase Description	Pre-Requisites	Step No	Test Steps	Expected Result	Actual Result	Test Case Status	Defect ID	Requirement Id	Comments

Test Scenario Vs Test Cases

Test Case	Test Scenario
A test case is a sequence of actions that are carried out to ensure that specific software features are working properly.	Any feature that may be tested is referred to as a test scenario.
A software testing approach that requires evaluating every possible data combination, benefits from test cases.	A test scenario is more flexible and focuses on the software's end-to-end functioning.
A test case considers what to test as well as how to test it.	A test scenario focuses solely on what to test.
For test execution, a test case necessitates more resources and time than a test scenario.	Test Scenario coverage is achieved by executing Test cases.
A test case contains test steps, expected results, and data.	A test scenario merely contains the functionality to be tested

Categories of Testing Techniques

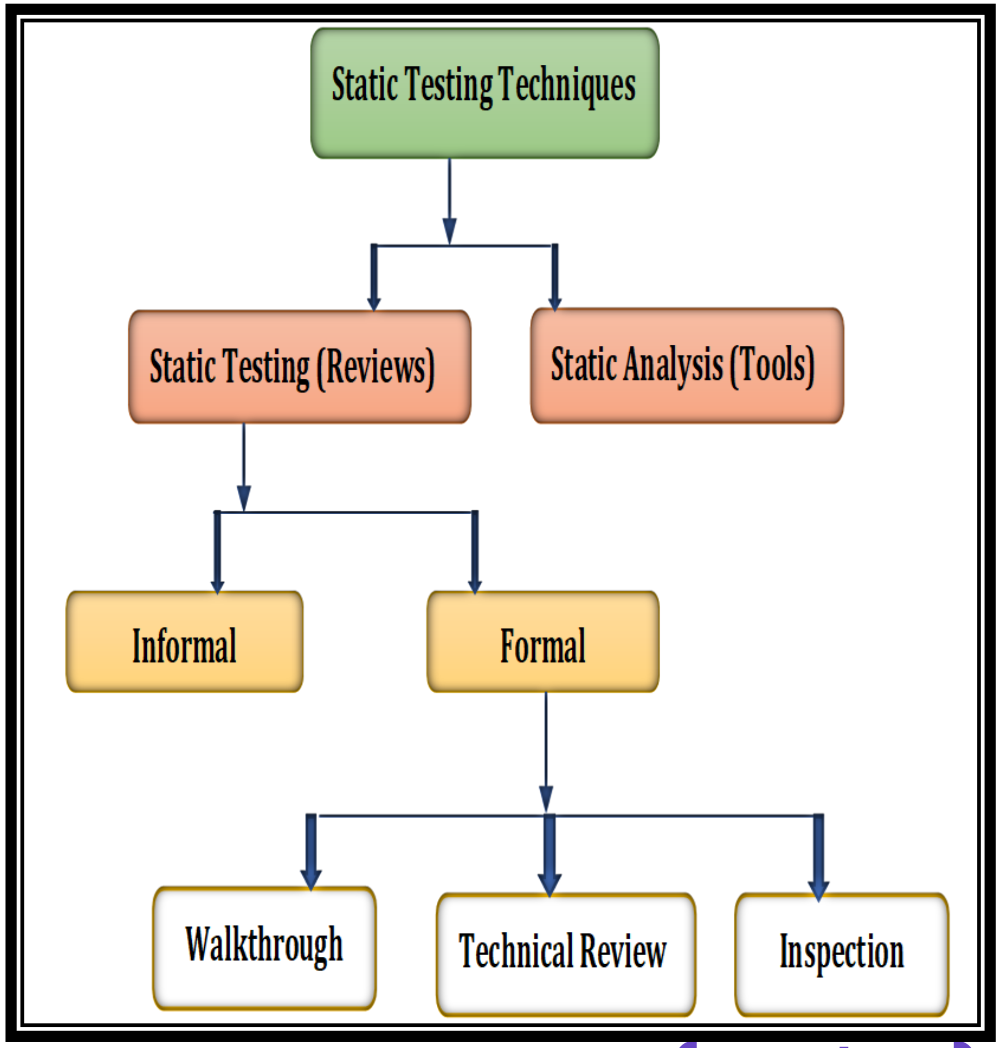
Static Testing Techniques

Static techniques and the test process

- **Static testing** relies on the **manual examination** of work products (i.e., reviews) or **tool-driven evaluation of the code** or other work products (i.e., static analysis).
- **Static analysis** is important for **safety-critical computer** systems (e.g., aviation, medical, or nuclear software), but static analysis has also become important and common in other settings.

Under Static Design Techniques, we have:

- **Static techniques and the test process**
- **Review process**
- **Static analysis by tools**



Static techniques and the test process

- Any software work product can be reviewed, including **requirements specifications, design specifications, code, test plans, test specifications, test cases, test scripts, user guides** or **web pages.**
- Compared to dynamic testing, **static techniques find causes of failures (defects) rather than the failures** themselves. Static analysis is an important part of **security testing**.
- So, basically static testing deals with **Quality Assurance**, involving **reviewing and auditing of code** and **other design documents**.

The various static test design techniques can be further divided into two parts :

- Static testing performed manually
- Static testing using tools.

Benefits of Static Testing

- Static testing enables the early detection of defects before dynamic testing is performed.
- Defects found early are often much cheaper to remove than defects found later in the lifecycle.
- Identifying defects which are not easily found by dynamic testing .
- Increasing development productivity (e.g., due to improved design, more maintainable code)
- Reducing development and testing cost and time .
- Improving communication between team members while participating in reviews.
- Finding and fixing defects during static testing is much cheaper than using dynamic testing to find defects and then fix them.

Work Products that Can Be Examined by Static Testing

- Specifications, including business requirements, functional requirements, and security requirements .
- Epics, user stories, and acceptance criteria.
- Architecture and design specifications .
- Code.
- Test ware, including test plans, test cases, test procedures, and automated test scripts.
- User guides.
- Web pages.
- Contracts, project plans, schedules, and budget planning .
- Configuration set up and infrastructure set up .

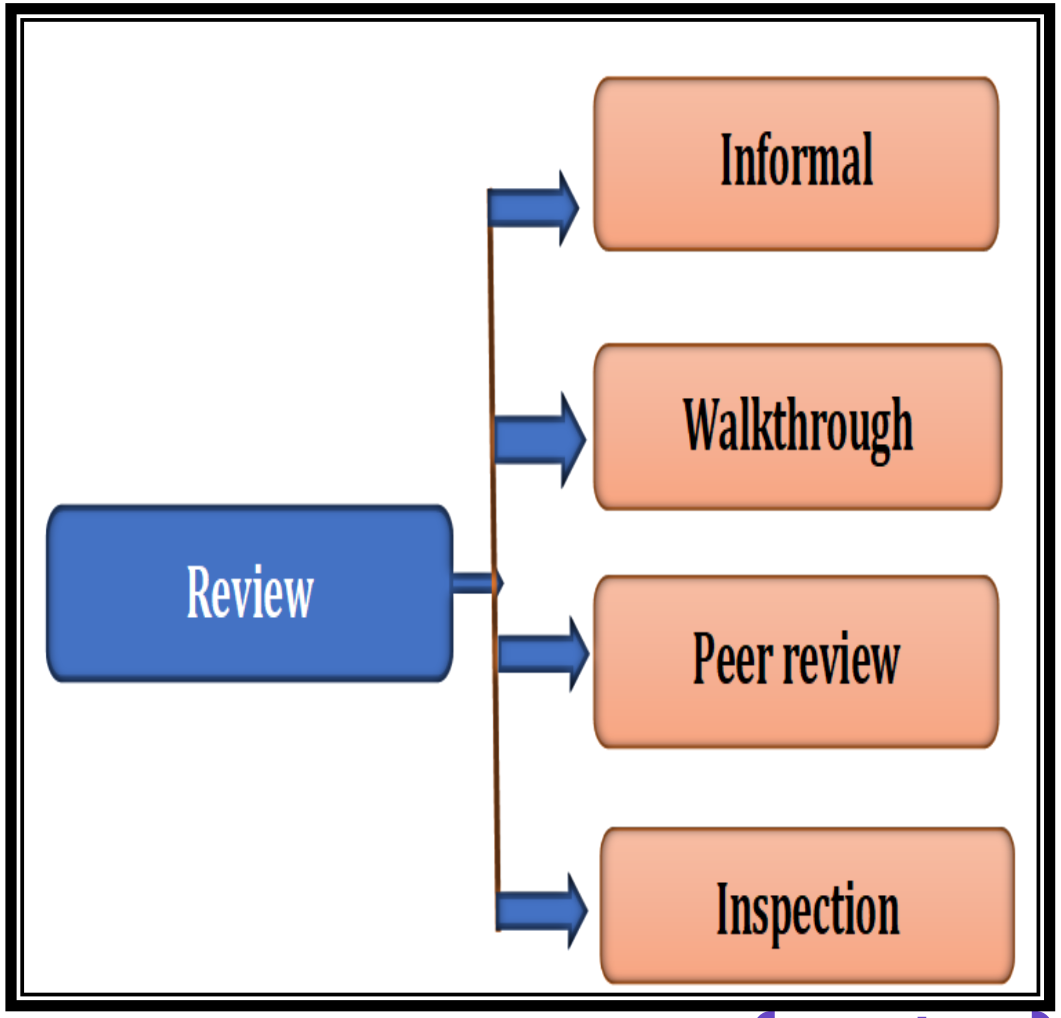
Review Process

Manual Static Design Techniques/Review Process

- Reviews are done in order to find the defects, issues, and ambiguities in the documents like requirements, design, etc.
- Reviews play an important role in static testing as it is better to find the cause of failure in the starting rather than failures at the end.
- As most of the issues arose after the development of software regarding the requirements, design not fixed in the starting or any ambiguity found in the documents.
- Reviews can be formal/informal depending on the stage of software testing.

Manual Static Design Techniques/Review Process

- **Informal reviews** are characterized by not following a **defined process and not having formal documented output**.
- **Formal reviews** are characterized by **team participation, documented results of the review, and documented procedures** for conducting the review.
- The focus of a review depends on the agreed objectives of the review (e.g**., finding defects, gaining understanding, educating participants such as testers and new team members, or discussing** and deciding by consensus).



Work Product Review Process

The review process comprises the following main activities:

Planning:

- Defining the scope, which includes the purpose of the review, what documents or parts of documents to review, and the quality characteristics to be evaluated.
- Estimating effort and timeframe .
- Identifying review characteristics such as the review type with roles, activities, and checklists .
- Selecting the people to participate in the review and allocating roles.
- Defining the entry and exit criteria for more formal review types (e.g., inspections)
- Checking that entry criteria are met (for more formal review types).

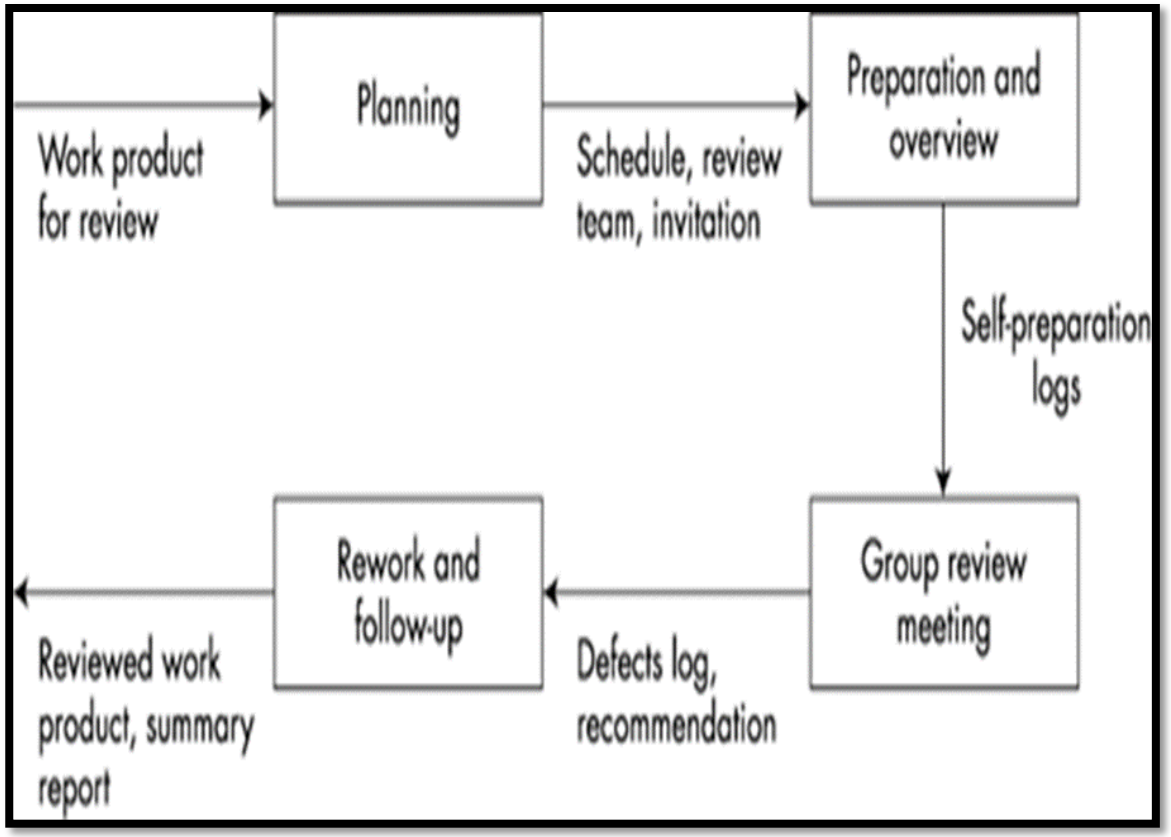
Work Product Review Process

Initiate review:

- Distributing the work product (physically or by electronic means) and other material, such as issue log forms, checklists, and related work products .
- Explaining the scope, objectives, process, roles, and work products to the participants.
- Answering any questions that participants may have about the review

Individual review or preparation:

- Reviewing all or part of the work product.
- Noting potential defects, recommendations, and questions.



Work Product Review Process

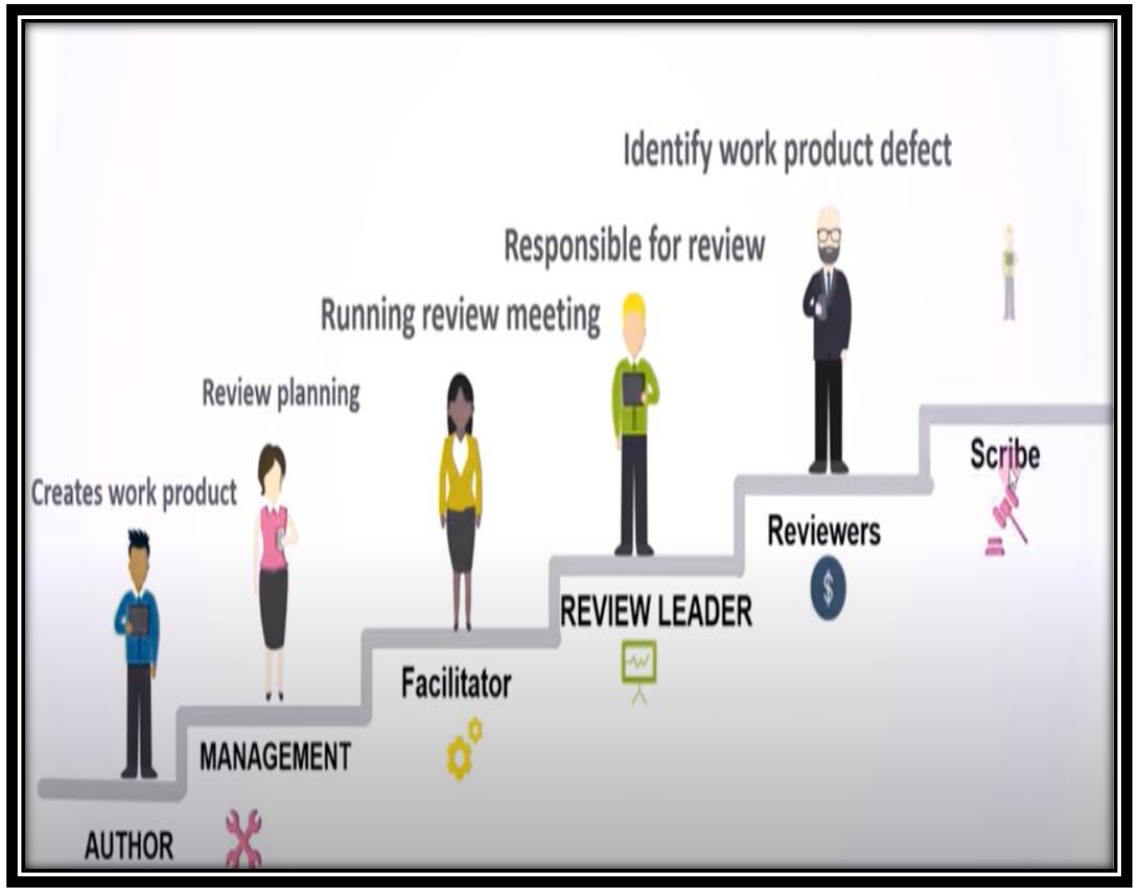
Issue communication and analysis :

- Communicating identified potential defects (e.g., in a review meeting)
- Analyzing potential defects, assigning ownership and status to them.
- Evaluating and documenting quality characteristics
- Evaluating the review findings against the exit criteria to make a review decision (reject; major changes needed; accept, possibly with minor changes)

Fixing and reporting:

- Creating defect reports for those findings that require changes to a work product
- Fixing defects found (typically done by the author) in the work product reviewed
- Communicating defects to the appropriate person or team.

Roles and responsibilities in a Formal Review



Author :

- Creates the work product under review.
- Fixes defects in the work product under review (if necessary).

Management:

- Is responsible for review planning.
- Decides on the execution of reviews.
- Assigns staff, budget, and time.
- Monitors ongoing cost-effectiveness.
- Executes control decisions in the event of inadequate outcomes.

Roles and responsibilities in a Formal review

Facilitator (often called moderator) :

- Ensures effective running of review meetings (when held).
- Mediates, if necessary, between the various points of view.
- Is often the person upon whom the success of the review depends.

Review leader:

- Takes overall responsibility for the review .
- Decides who will be involved and organizes when and where it will take place.

Roles and responsibilities in a Formal review

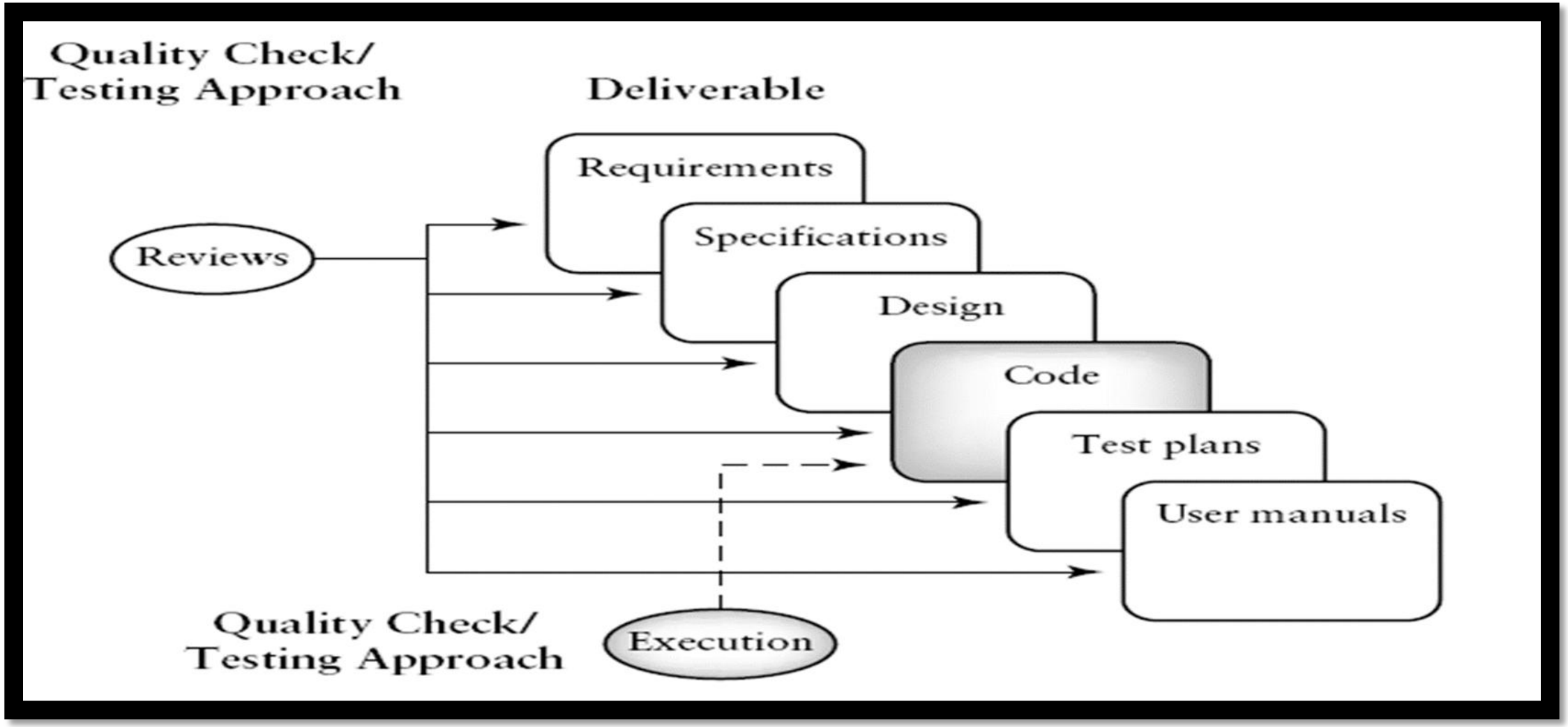
Reviewers:

- May be subject matter experts, persons working on the project, stakeholders with an interest in the work product, and/or individuals with specific technical or business backgrounds.
- May represent different perspectives (e.g., tester, developer, user, operator, business analyst, usability expert, etc.)

Scribe (or recorder):

- Collates potential defects found during the individual review activity.
- Records new potential defects, open points, and decisions from the review meeting (when held) .

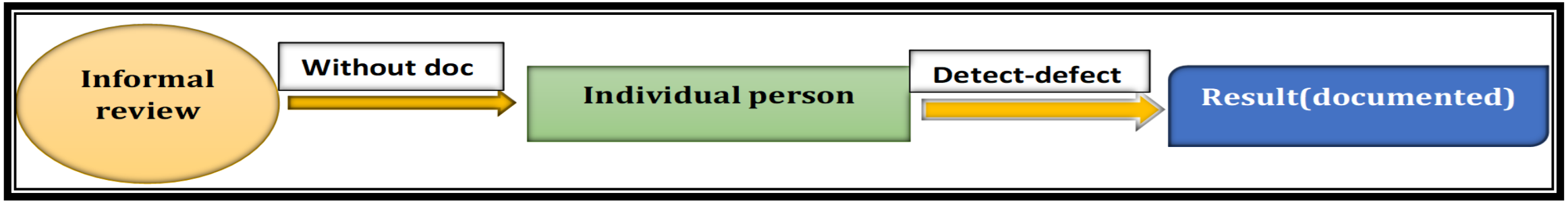
Roles and responsibilities in a Formal review



Review Types:

Informal Reviews

- As the name suggests, an informal review done by an individual without any process or documentation.
- Main purpose: detecting potential defects.
- Not based on a formal (documented) process.
- May not involve a review meeting.
- Results may be documented.



Review Types:

Walkthrough

- A Walk-through is a step-by-step presentation of different requirements and design documents by their authors.
- This is done with the intent of finding defects or any missing pieces in the documents.
- Main purposes: find defects, improve the software product, consider alternative implementations, evaluate conformance to standards and specifications.
- Review meeting is typically led by the author of the work product.
- Scribe is mandatory .
- Potential defect logs and review reports are produced.

Review Types:

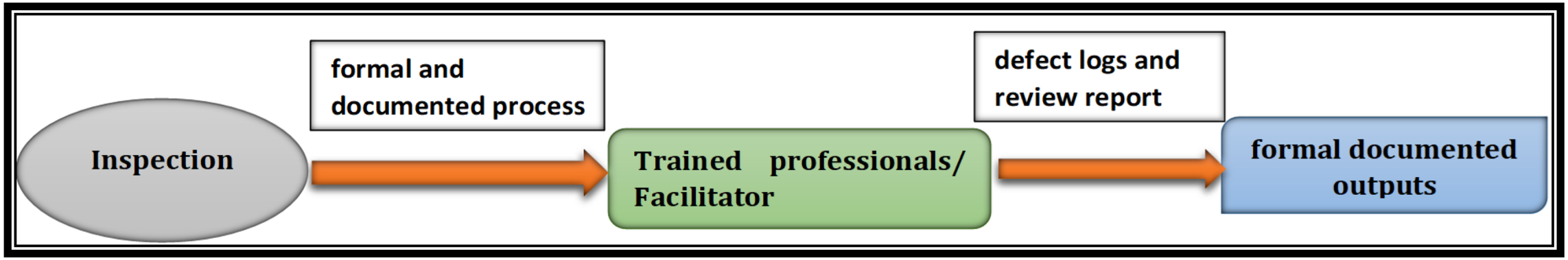
Technical Review

- A technical review involves reviewing the technical approach used during the development process.
- Main purposes: gaining consensus, detecting potential defects.
- Individual preparation before the review meeting is required.
- Scribe is mandatory, ideally not the author .
- Reviewers should be technical peers of the author, and technical experts in the same or other disciplines

Review Types:

Inspection

- An inspection is a formal and documented process of reviewing the different documents by experts or trained professionals.
- Main purposes: detecting potential defects.
- Follows a defined process with formal documented outputs, based on rules and checklists.
- Scribe is mandatory
- Review meeting is led by a trained facilitator.
- Potential defect logs and review report are produced.



Quiz

1.Which of the following statements CORRECTLY reflects the value of static testing?

a) By introducing reviews, we have found that both the quality of specifications and the time required for development and testing have increased

b) Using static testing means we have better control and cheaper defect management due to the ease of detecting defects later in the lifecycle.

c) Now that we require the use of static analysis, missed requirements have decreased and communication between testers and developers has improved

d) Since we started using static analysis, we find coding defects that might have not been found by performing only dynamic testing

Quiz

2. Which of the following statements on the use of checklists in a formal review is CORRECT?

a) As part of the review planning, the reviewers create the checklists needed for the review

b) As part of the issue communication, the reviewers fill in the checklists provided for the review

c) As part of the review meeting, the reviewers create defect reports based on the checklists provided for the review

d) As part of the review initiation, the reviewers receive the checklists needed for the review

Quiz

3. Which of the following CORRECTLY matches the roles and responsibilities in a formal review?

a) Manager – Decides on the execution of reviews

b) Review Leader - Ensures effective running of review meetings

c) Scribe – Fixes defects in the work product under review

d) Moderator – Monitors ongoing cost-effectiveness

Quiz

4. In a formal review, what is the role name for the participant who runs an inspection meeting?

a) Facilitator

b) Programmer

c) Author

d) Project Manager

Quiz

5.You are reading a user story in the product backlog to prepare for a meeting with the product owner and a developer, noting potential defects as you go. Which of the following statements is true about this activity?

a) It is not a static test, because static testing involves execution of the test object

b) It is not a static test, because static testing is always performed using a tool

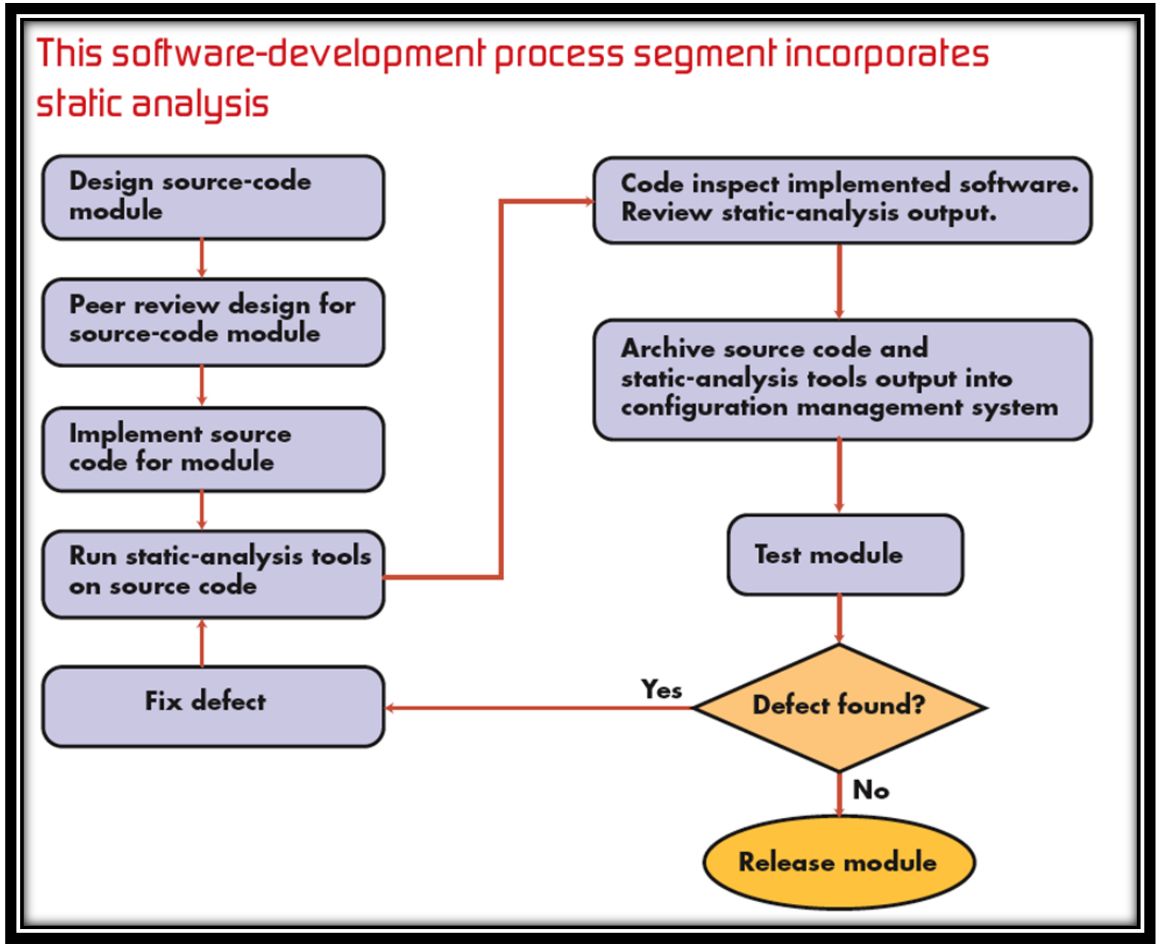
c) It is a static test, because any defects you find could be found cheaper during dynamic testing.

d) It is a static test, because static testing does not involve execution of the test object

Static Analysis by Tools

About Static analysis by tools

- The objective of static analysis is to find defects in software source code and software models.
- Static analysis can locate defects that are hard to find in testing. As with reviews, static analysis finds defects rather than failures.
- Static analysis tools analyze program code (e.g., control flow and data flow), as well as generated output such as HTML and XML.



Static Design Techniques Using Tools

The static analysis techniques for the source code evaluation using tools are:

- **Control flow analysis** – The control flow analysis requires analysis of all possible control flows or paths in the code.
- **Data flow analysis** – The data flow analysis requires the analysis of data in the application and its different states
- **Compliance with coding standards** – This evaluates the compliance of the code with the different coding standards.
- **Analysis of code metrics** – The tool used for static analysis is required to evaluate the different metrics like lines of code, complexity, code coverage, etc.

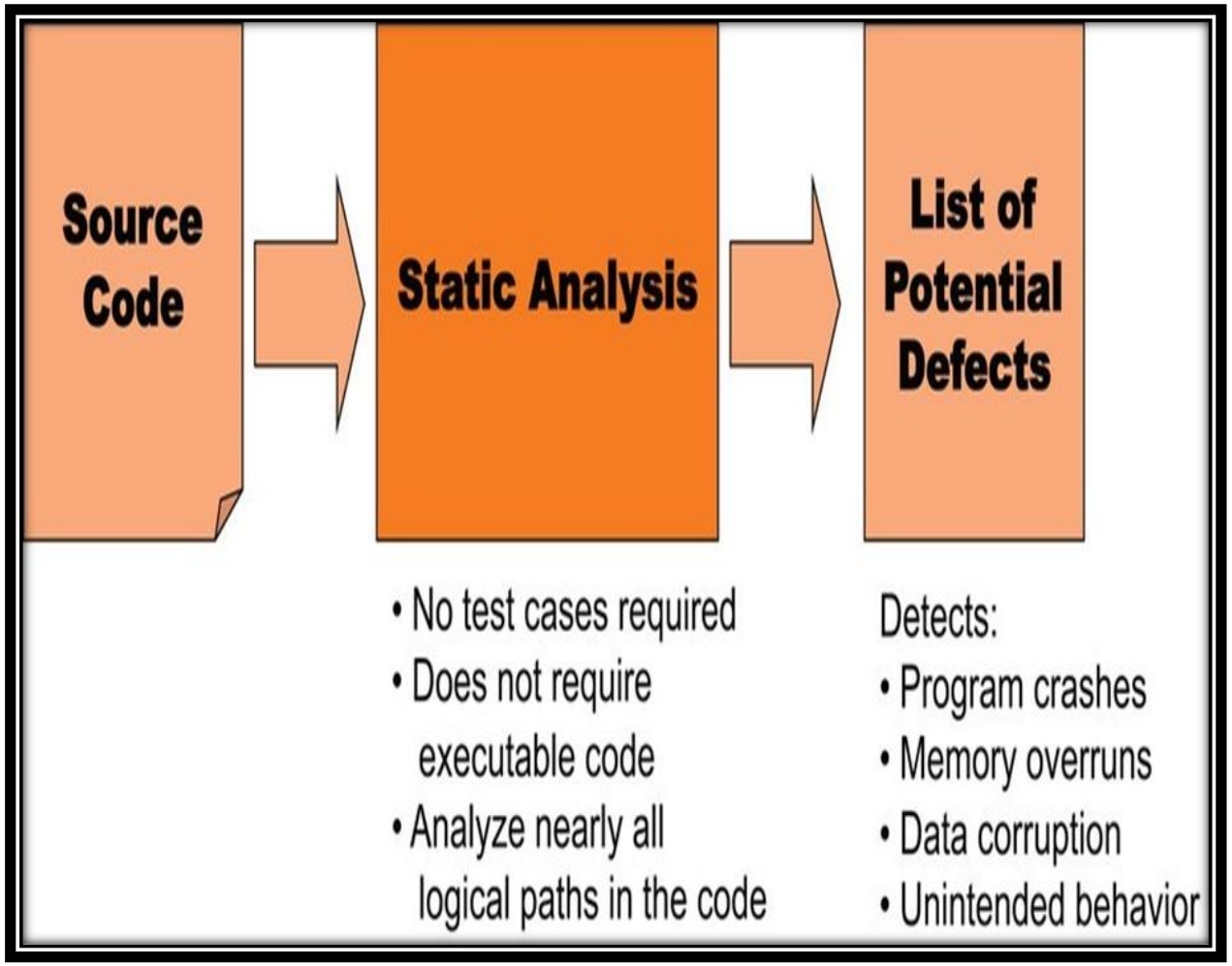
Value of Static analysis

- Early detection of defects prior to test execution.
- Early warning about suspicious aspects of the code or design, by the calculation of metrics, such as a high complexity measure.
- Identification of defects not easily found by dynamic testing.
- Detecting dependencies and inconsistencies in software models.
- Improved maintainability of code and design.
- Prevention of defects, if lessons are learned in development.

Advantages of static analysis (Image by author)

Typical defects discovered by static analysis tools

- Referencing a variable with an undefined value.
- Inconsistent interface between modules and components.
- Variables that are never used.
- Unreachable (dead) code.
- Programming standards violations.
- Security vulnerabilities.
- Syntax violations of code and software models.



Quiz

1. During a period of intensive project overtime, a system architecture document is sent to various project participants, announcing a previously-unplanned technical review to occur in one week. No adjustments are made to the participants' list of assigned tasks. Based on this information alone, which of the following is a factor for review success that is MISSING?

a) Appropriate review type

b) Adequate time to prepare

c) Sufficient metrics to evaluate the author

d) Well-managed review meeting

Ans: b)

Quiz

2.You are working as a tester on an Agile team and have participated in over two dozen user story refinement sessions with the product owner and the developers on the team at the start of each iteration. As the reviews have gotten more effective at detecting defects in user stories and the product owner more adept at correcting those defects, you and the team notice that the team's velocity, as shown in your burndown charts, has started to increase. Which of the following is a benefit of static testing that MOST DIRECTLY applies to increased velocity?

a) Increasing total cost of quality

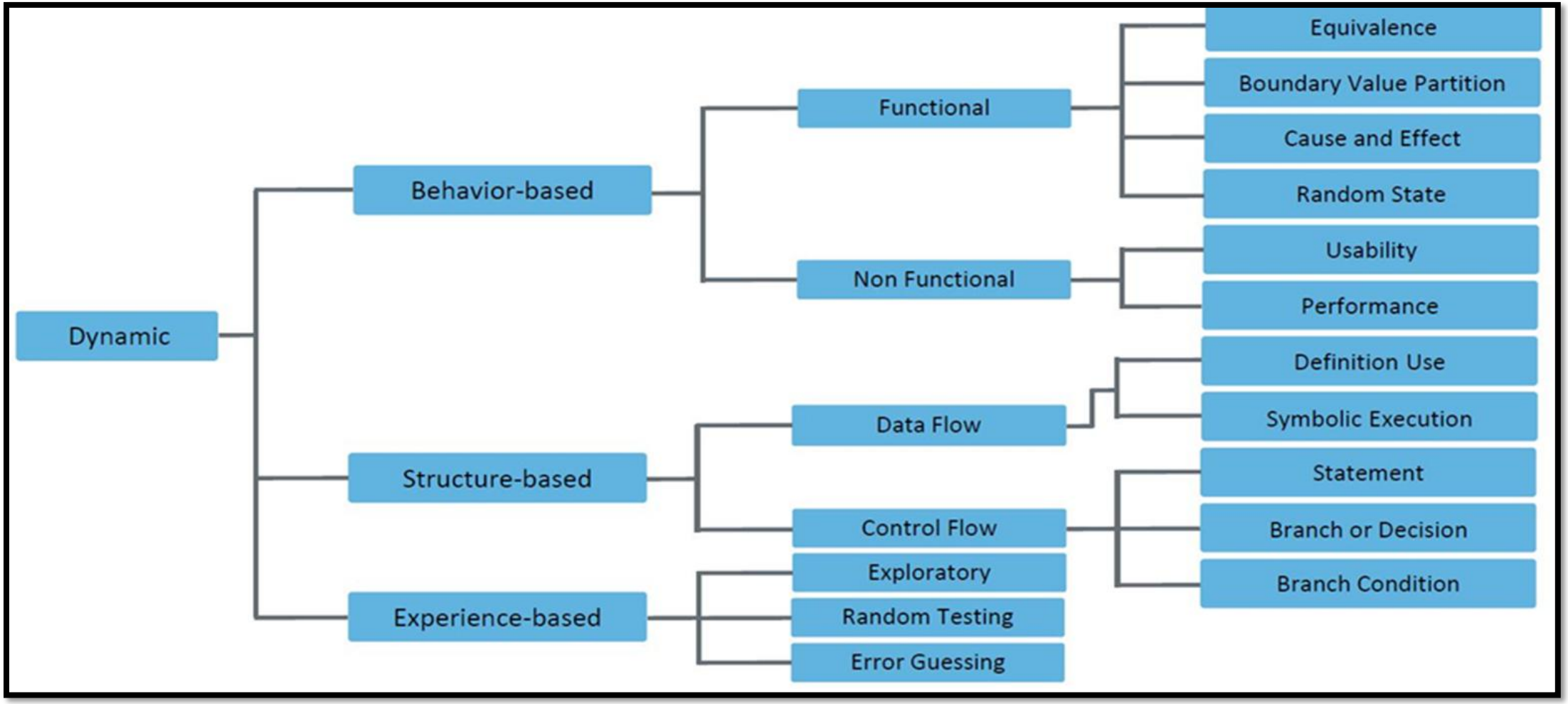
b) Reducing testing cost

c) Increasing development productivity

d) Reducing total cost of quality

Ans: c)

Overview of Testing Techniques



Black-box Techniques

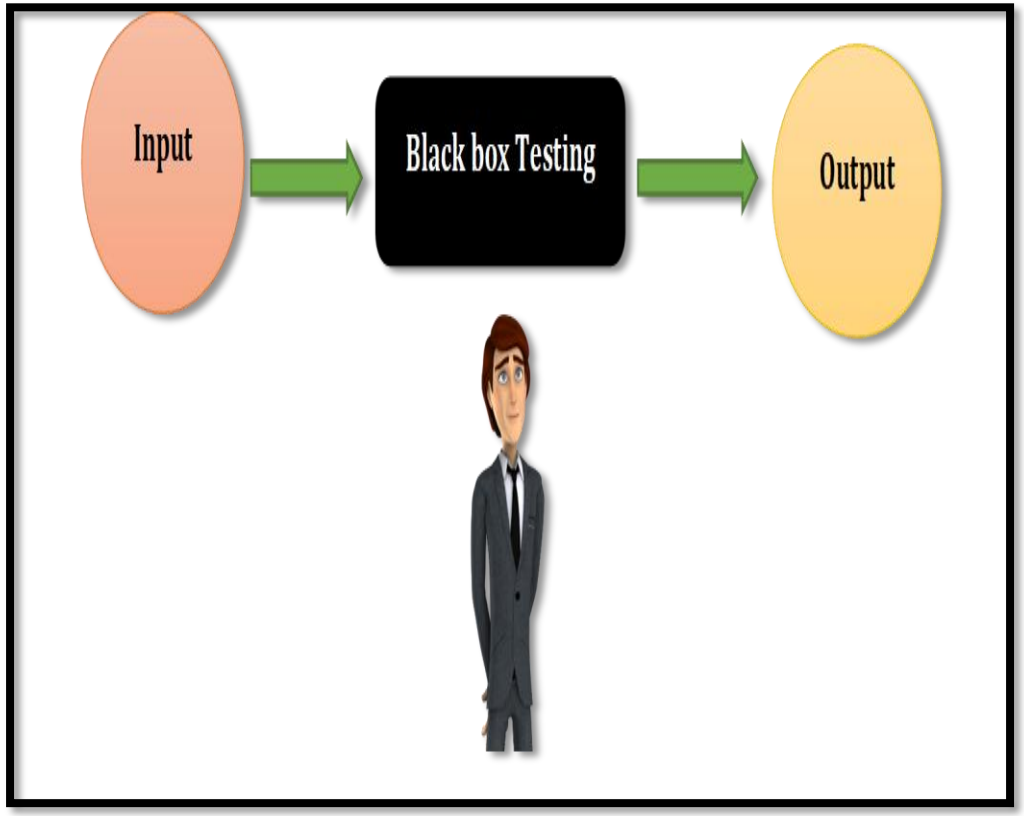
Test techniques are classified as ,

- **Black-box test techniques** - (also called **behavioral or behavior-based** techniques) are based on an analysis of the **appropriate test basis (e.g., formal requirements documents, specifications, use cases, user stories, or business processes)**. These techniques are applicable to both **functional and non-functional testing.**
- Black-box test techniques **concentrate on the inputs and outputs of the test object without reference to its internal structure.**

Black-box Techniques

- The **primary source** of black box testing is a **specification of requirements** that is stated by the **customer.**
- In this method, **tester selects a function and gives input value to examine its functionality**, and **checks whether the function is giving expected output or not**. If the function produces correct output, then it is passed in testing, otherwise failed.

Black-box Techniques



Common characteristics of black-box test techniques include the following:

- Test conditions, test cases, and test data are derived from a test basis that may include software requirements, specifications, use cases, and user stories.
- Test cases may be used to detect gaps between the requirements and the implementation of the requirements, as well as deviations from the requirements.
- Coverage is measured based on the items tested in the test basis and the technique applied to the test basis

White-box Techniques

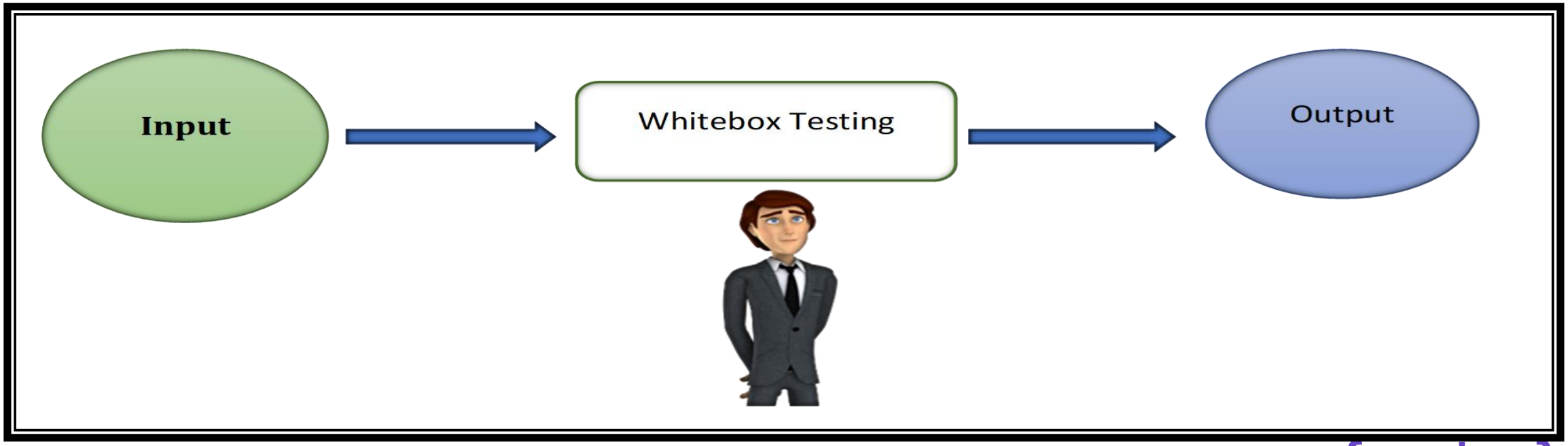
- **White-box test techniques** -(also called **structural or structure-based techniques**) are based on an **analysis of the architecture, detailed design, internal structure, or the code of the test object**.

Common characteristics of white-box test techniques include:

- **Test conditions, test cases, and test data** are derived from a test basis that may include **code, software architecture, detailed design,** or any other source of information regarding the structure of the software .
- **Coverage** is measured based on the items tested within a **selected structure** (e.g., the code or interfaces) and the technique applied to the test basis .

White-box Techniques

- The primary goal of white box testing is to focus on the **flow of inputs and outputs** through the software and strengthening the security of the software.
- The term "Whitebox" was used because of the **see-through box concept**.



Experienced-based Test Techniques

- **Experience-based test techniques** - leverage the **experience of developers, testers and users to design, implement, and execute tests**. These techniques are often combined **with black-box and white-box test techniques.**
- The experience-based testing technique is based on the **skill and experience of the testers, experts, users etc.,**
- It is conducted in an **Ad-hoc manner** because proper specifications are not available to test the applications.

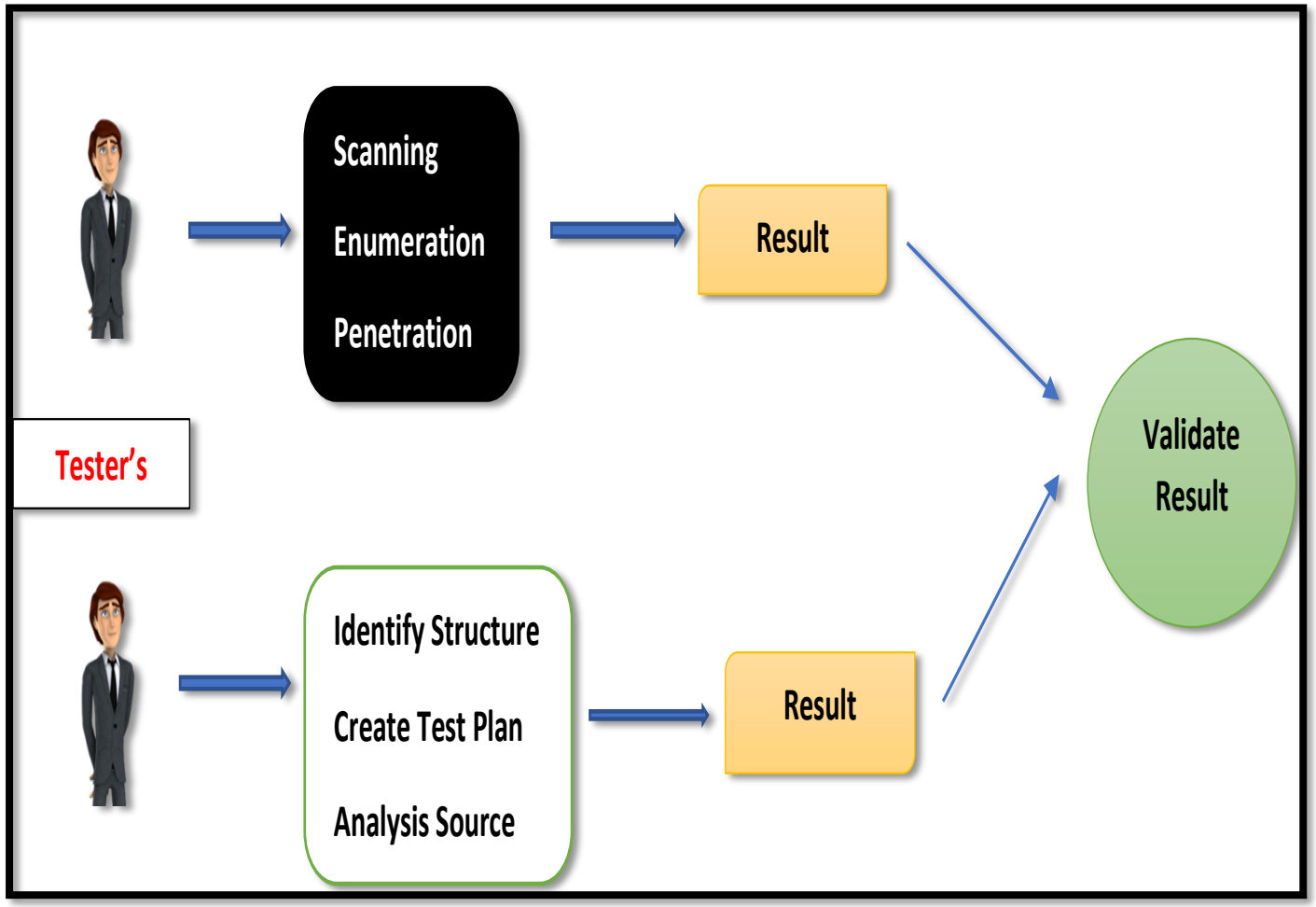
Experienced-based Test Techniques

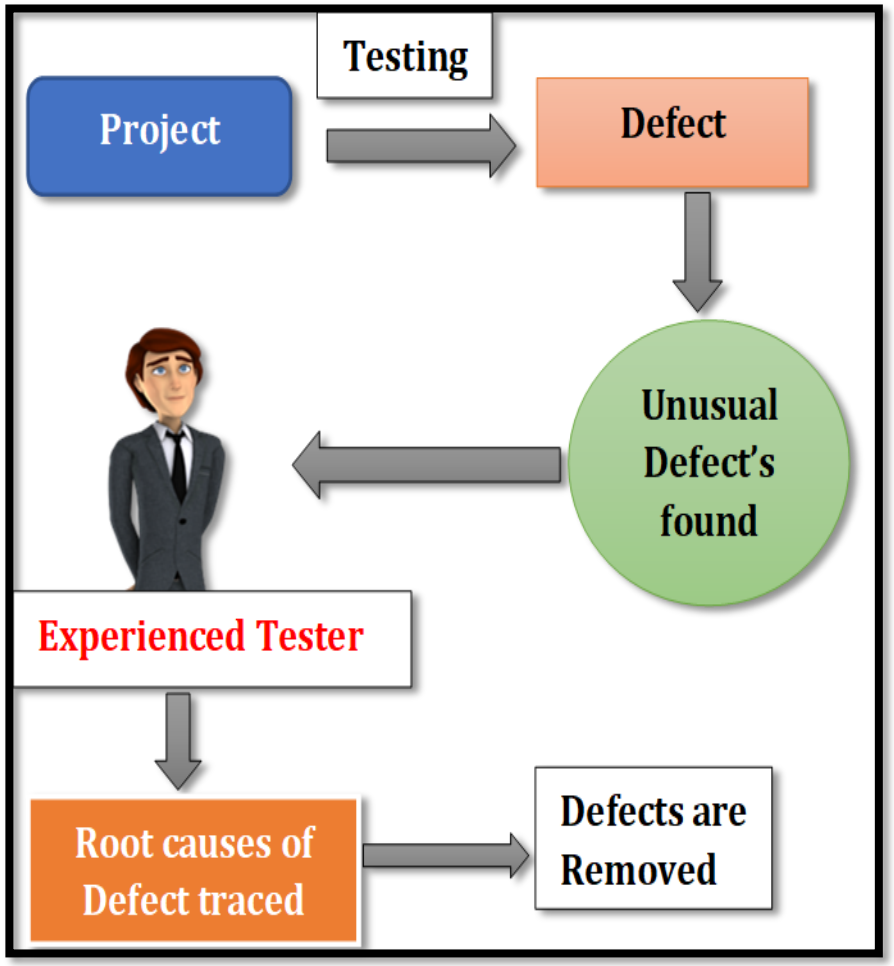
- This technique is used for low-risk system. This kind of **testing is done even when there is no specifications or have inadequate specification list.**

Common characteristics of experience-based test techniques include:

- Test conditions, test cases, and test data are derived from a test basis that may include knowledge and experience of testers, developers, users and other stakeholders.

Experienced-based Test Techniques





White-box Testing Techniques

White-box Testing

- White-box testing is based on the **internal structure of the test object**. White-box test techniques can be used at all test levels.

Most used techniques in White-box Testing are,

- **Statement Testing and Coverage**.
- **Decision Testing and Coverage**.
- **Branch Coverage**.

White-box Testing

Other techniques used in White-box Testing are,

- Condition Coverage
- Multiple Condition Coverage
- Finite State Machine Coverage
- Path Coverage
- Control flow testing
- Data flow testing

Statement Coverage (or) Statement Testing

- Statement coverage type of white box software testing technique. It is called line or segment coverage testing.
- This testing is done **to check whether the code written is qualitative** or **whether it does what it is expected to be done**. It is checked in true conditions it is verified for every line of statements.

Consider an example:

```
Read A
Read B
If A > B then
x = 0
End if
```

Statement Coverage (or) Statement Testing

Statement coverage=(no.of.executed statements / total.no.of statements)x100%

- To achieve 100% statement coverage, we need to check only one test case here.
- We ensure that A value is greater than B for input values like A=14 and B=11.
- Here we check the structural design first as we are choosing input values to make certain the statement of coverage.
- The main disadvantage of this technique is here we cannot check the false condition of statements.

Decision Coverage (or) Decision Testing

- The decision testing is a process which checks the behavior of the application by providing different combinations of input and obtains various results.
- Decision coverage covers all possible outcomes of each Boolean condition of the code by using control flow graph or chart.
- It is also known as **branch coverage or all-edges coverage or edge testing .**

Decision Coverage = (No of Decision outcome exercised) / (total No of Decision Outcomes) *100%

Decision Coverage (or) Decision Testing

For E.g.,

```
display(int a) {
    If (a> 5){
        a=a*3;
         }
    Print (a);
   }
```

1.input a=3 output=3

2. input a=6 output=18

Test Case	Value of A	output	Decision Coverage
1	3(False condition)	3	50%
2	6(True Condition)	18	50%

NOTE: Draw a flow-chart diagram by yourself .

Quiz

1. Which of the following descriptions of statement coverage is CORRECT ?

a) Statement coverage is a measure of the number of lines of source code exercised by tests.

b) Statement coverage is a measure of the proportion of executable statements in the source code exercised by tests.

c) Statement coverage is a measure of the percentage of lines of source code (without comments) exercised by tests.

d) Statement coverage is a measure of the number of executable statements in the source code exercised by tests.

Ans: b)

Quiz

2. Which of the following descriptions of decision coverage is CORRECT?

a) Decision coverage is a measure of the percentage of possible paths through the source code exercised by tests.

b) Decision coverage is a measure of the percentage of business flows through the component exercised by tests.

c) Decision coverage is a measure of the 'if' statements in the code that are exercised with both the true and false outcomes.

d) Decision coverage is a measure of the proportion of decision outcomes in the source code exercised by tests.

Ans: d)

Quiz

3.The following statement refers to decision coverage:

"When the code contains only a single 'if' statement and no loops or CASE statements, and its

execution is not nested within the test, any single test case we run will result in 50% decision

Coverage ." Which of the following statement is correct?

a) The statement is true. Any single test case provides 100% statement coverage and therefore 50% decision coverage

b) The statement is true. Any single test case would cause the outcome of the "if" statement to be either true or false

c) The statement is false. A single test case can only guarantee 25% decision coverage in this case

d) The statement is false. The statement is too broad. It may be correct or not, depending on the tested software

Ans: b)

Quiz

4. Which one of the following is the description of statement coverage?

a) It is a metric, which is the percentage of test cases that have been executed

b) It is a metric, which is the percentage of statements in the source code that have been executed

c) It is a metric, which is the number of statements in the source code that have been executed by test cases that are passed.

d) It is a metric, that gives a true/false confirmation if all statements are covered or not

Ans: b)

Quiz

5. Which statement about the relationship between statement coverage and decision coverage is true?

a) 100% decision coverage also guarantees 100% statement coverage

b) 100% statement coverage also guarantees 100% decision coverage

c) 50% decision coverage also guarantees 50% statement coverage.

d) Decision coverage can never reach 100%

Ans: a)

Quiz

1) Which of the following would structure-based test design techniques be most likely to be applied to?

1. **Boundaries between mortgage interest rate bands**
2. **An invalid transition between two different arrear statuses**
3. **The business process flow for mortgage approval**
4. **Control flow of program to calculate repayments**

a) 2, 3 and 4

b) 2 and 4

c) 3 and 4

d) 1, 2 and 3

Answer : Option c)

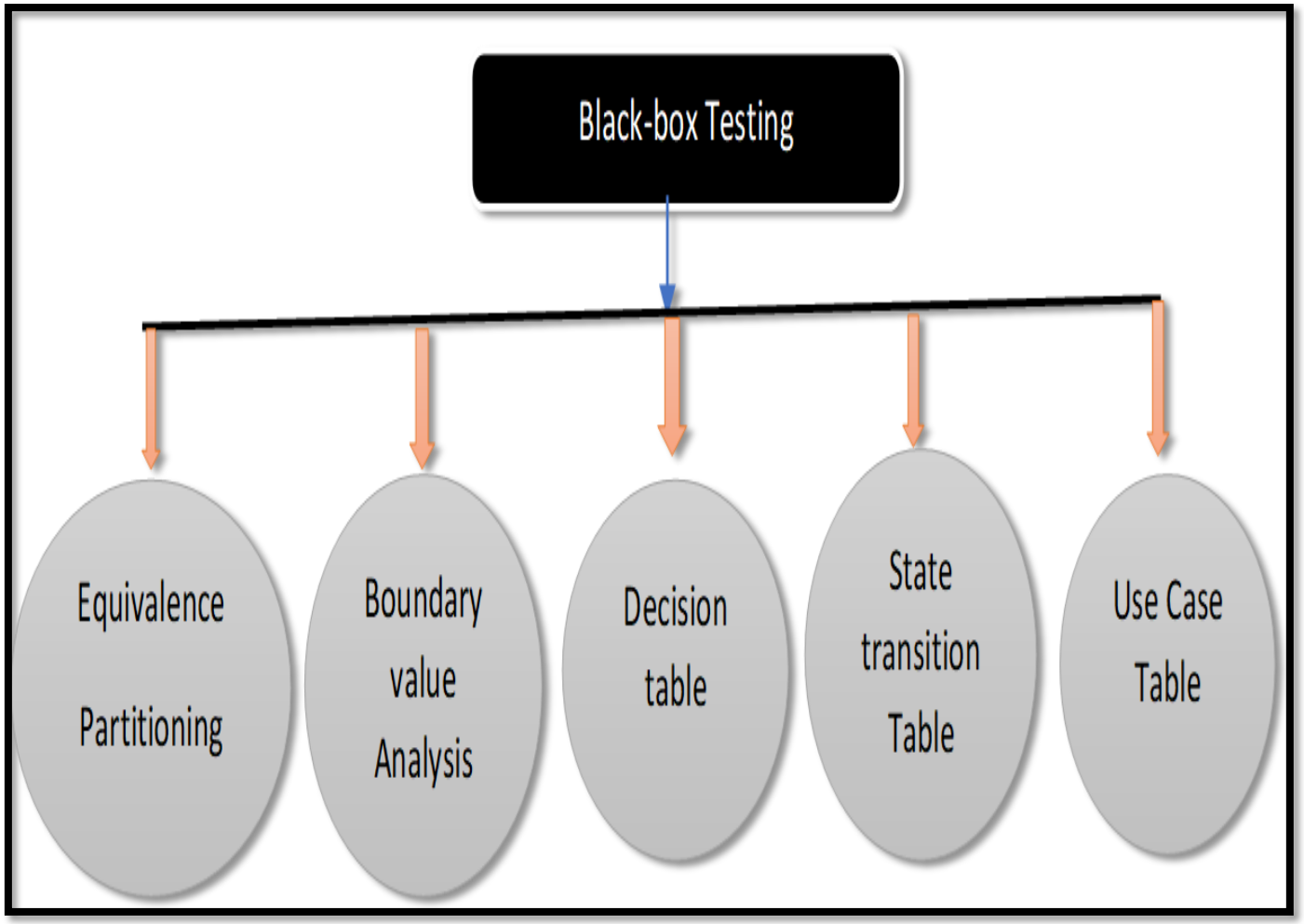
Black-box Testing Techniques

Black-box Testing

- Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding.

Techniques used in Black-box Testing are,

- **Equivalence Partitioning**
- **Boundary Value Analysis**
- **Decision Table Testing**
- **State Transition Testing**
- **Use Case Testing**



Equivalence Partitioning

- **Equivalence partitioning divides data into partitions (also known as equivalence classes)** in such a way that all the members of a given partition are expected to be processed in the same way. There are equivalence partitions for both valid and invalid values.
- Valid values are values that should be accepted by the component or system. An equivalence partition containing valid values is called a **"valid equivalence partition."**
- Invalid values are values that should be rejected by the component or system. An equivalence partition containing invalid values is called an **"invalid equivalence partition."**
- Partitions can be identified for any data element related to the test object, including inputs, outputs, internal values, time-related values (e.g., before or after an event) and for interface parameters (e.g., integrated components being tested during integration testing).

Equivalence Partitioning

- Any partition may be divided into sub partitions if required.
- Each value must belong to one and only one equivalence partition.
- When invalid equivalence partitions are used in test cases, they should be tested individually.

For example:

- Assume

EQUIVALENCE PARTITIONING		
Invalid	**Valid**	**Invalid**
<=17	18-56	>=57

Equivalence Partitioning

Possible values :

Valid Input: **18 – 56**

Invalid Input: less than or equal to 17 **(<=17)**, greater than or equal to 57 (**>=57**)

Valid Class: 18 – 56 = **Pick any one input test data from 18 – 56**

Invalid Class 1: <=17 = **Pick any one input test data less than or equal to 17**

Invalid Class 2: >=57 = **Pick any one input test data greater than or equal to 57**

We have one valid and two invalid conditions here.

Equivalence partitioning works as like above mentioned scenario.

Boundary Value Analysis

- Boundary value analysis (BVA) is an extension of equivalence partitioning but can only be used when the partition is ordered, consisting of numeric or sequential data.
- The basic idea in normal boundary value testing is to select input variable values at their:
  - **Minimum**
  - **Just above the minimum**
  - **A nominal value**
  - **Just below the maximum**
  - **Maximum**

Boundary Value Analysis

For example,

- We must test a field which accepts Age **18 – 56**.

BOUNDARY VALUE ANALYSIS		
Invalid (min -1)	**Valid (min, +min, -max, max)**	**Invalid (max +1)**
17	18, 19, 55, 56	57

Boundary Value Analysis

Minimum boundary value is 18

Maximum boundary value is 56

Valid Inputs: 18,19,55,56

Invalid Inputs: 17 and 57

Test case 1: Enter the value 17 (18-1) = Invalid

Test case 2: Enter the value 18 = Valid

Test case 3: Enter the value 19 (18+1) = Valid

Test case 4: Enter the value 55 (56-1) = Valid

Test case 5: Enter the value 56 = Valid

Test case 6: Enter the value 57 (56+1) =Invalid

Decision Table Testing

- Decision table technique is one of the most extensively used test case design strategies for black box testing is the decision table technique. This is a systematic methodology in which various input combinations and their corresponding system behavior are evaluated.
- It's also called as a cause-and-effect table. This technique is used to choose test cases in a systematic manner; it reduces testing time and ensures that the software application's testing area is adequately covered.
- For features with a logical relation between two or more inputs, the decision table technique is effective.

Decision Table Testing

- This technique involves correct input combination and determines the outcome of various input combinations. We use decision table technique to design the test cases.
- A logic of **2^n** formula is used to calculate the number of combinations to test with two different combinations **– Yes or No.**
- For example, As the number of inputs grows, the relevance of this strategy becomes apparent. 2^n, where n is the number of inputs, equals the number of possible combinations. The number of combinations for n = 10, which is 1024. Obviously, you cannot test all available combinations, but you can choose a large subset of them using the decision-based testing technique.

Decision Table Testing

- For any combination of Input the result will be **True/Yes – 1**. (Or)
- The result will be **False/No - 2.**

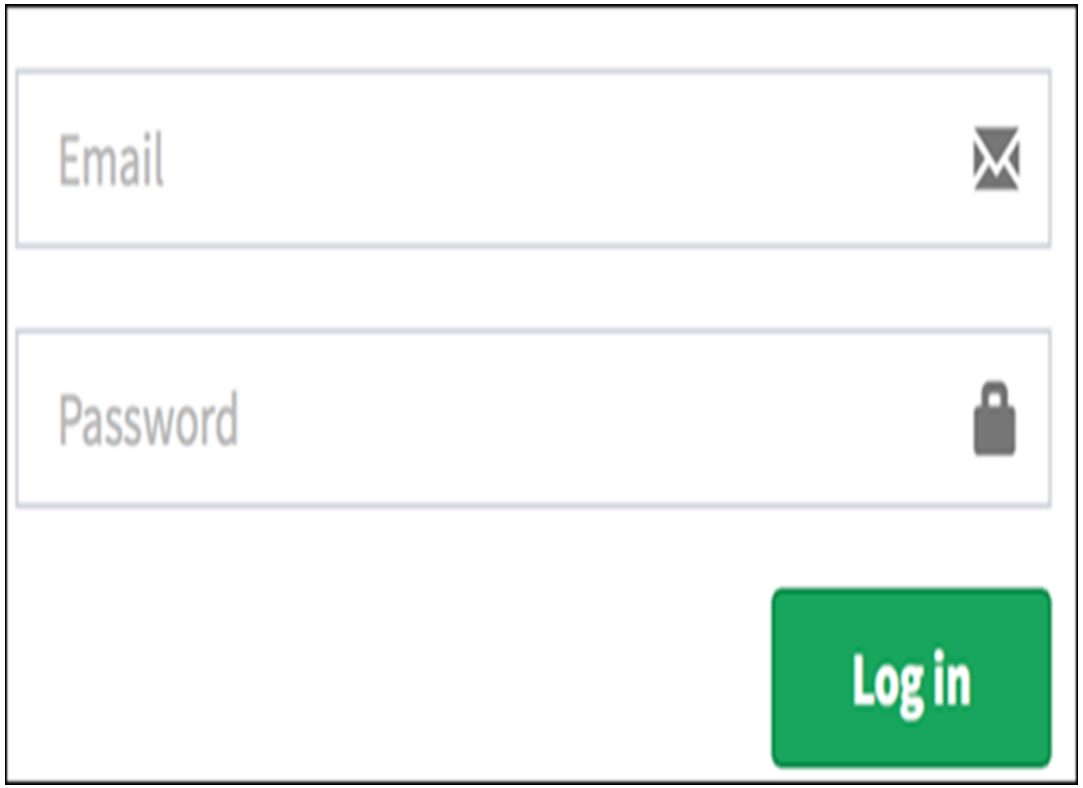
For conditions:

- Y means the condition is true (may also be shown as T or 1)
- N means the condition is false (may also be shown as F or 0)
- — means the value of the condition doesn’t matter (may also be shown as N/A)

For actions:

- X means the action should occur (may also be shown as Y or T or 1)
- Blank means the action should not occur (may also be shown as – or N or F or 0)

Decision Table Testing



For example,

Specification or Legend

- T – Correct username/password
- F – Wrong username/password
- E – Error message is displayed
- H – Home screen is displayed

Decision Table Testing

Test Case:

- Case 1 – Username and password both were wrong. The user is shown an error message.
- Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
- Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
- Case 4 – Username and password both were correct, and the user navigated to homepage.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Username (T/F)	F	T	F	T
Password (T/F)	F	F	T	T
Output (E/H)	E	E	E	H

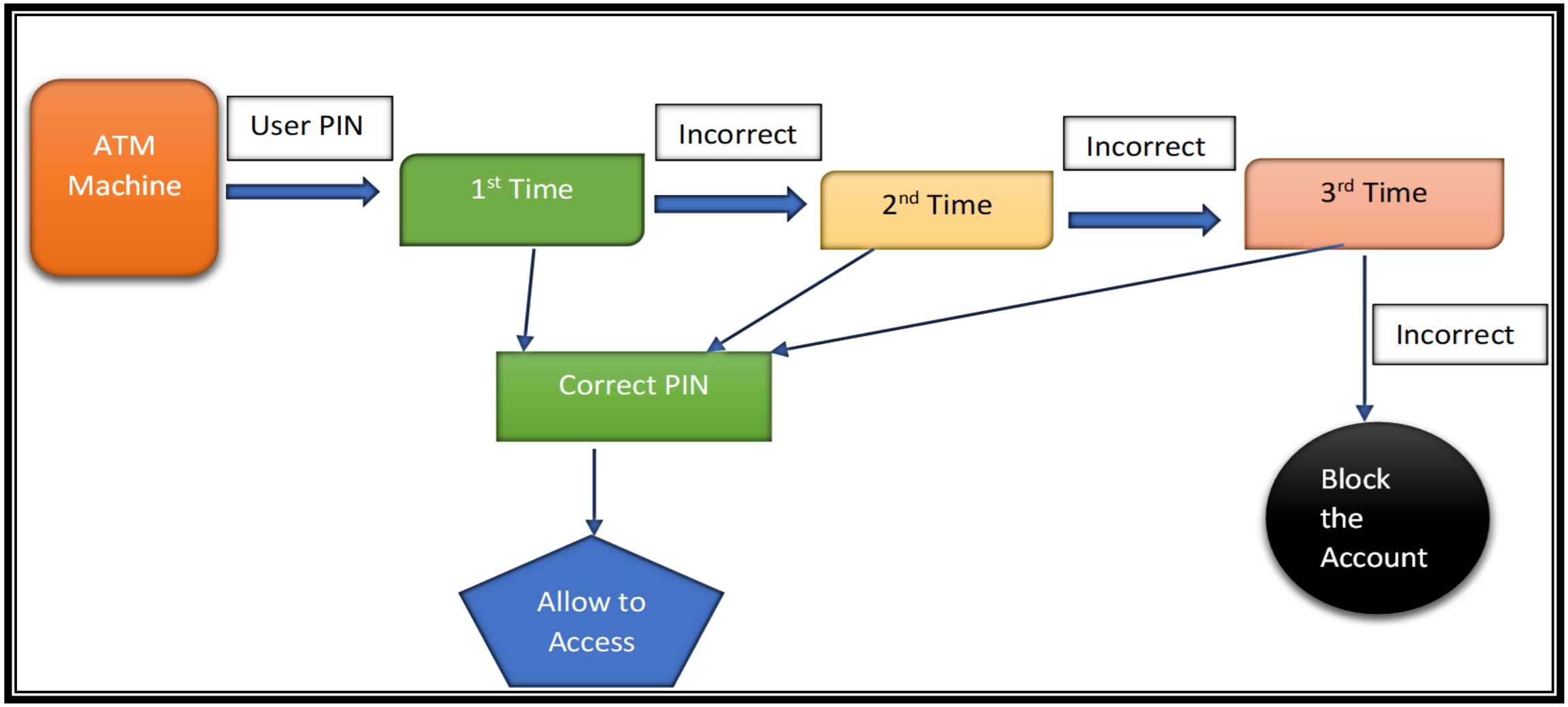
State Transition Testing

- State Transition Testing is a technique that is used to test **the different states of the system under test.**
- The state of the system changes depending upon the conditions or events.
- State transition testing is used for **menu-based applications** and is widely used within the embedded software industry.

For example,

- Let’s consider an ATM system function where if the user enters the invalid password three times the account will be locked. In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.

State Transition Testing



State Transition Testing

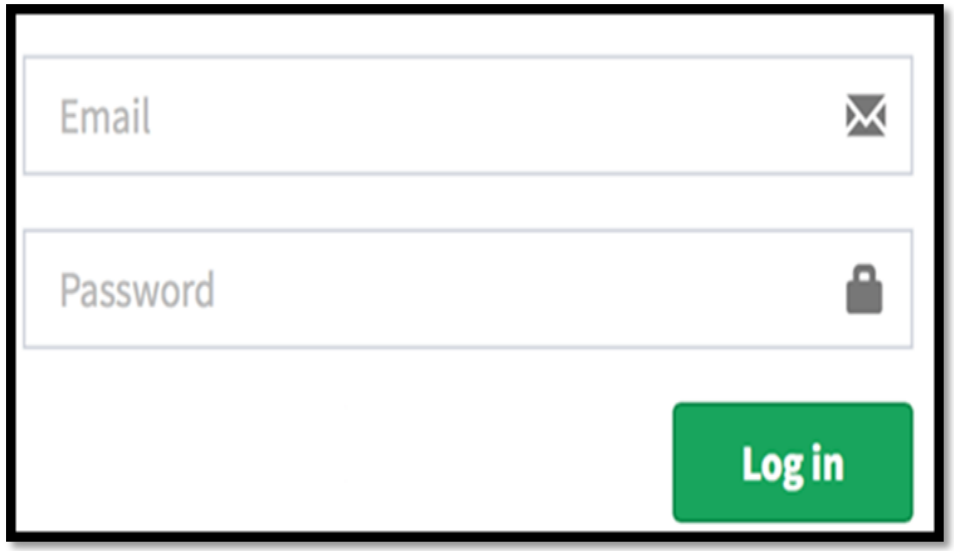
States	Correct Password (Events)	Incorrect Password (Events)
S1) Start	S5	S2
S2)1st try	S5	S3
S3)2nd try	S5	S4
S4)3rd try	S5	S6
S5) Access Granted	-	-
S6) Account block	-	-

Use Case Testing

- Tests can be derived from use cases, which are a specific way of designing interactions with software items.
- Use cases are associated with actors (human users, external hardware, or other components or systems) and subjects (the component or system to which the use case is applied).
- A use case can be described by interactions and activities, as well as preconditions, postconditions and natural language where appropriate.
- Interactions may be represented graphically by workflows, activity diagrams, or business process models.

Use Case Testing

For Example,



Specification of Success Scenario	Steps	Description
A: Actor S: System	1 2 3	A: Enter name and Password S: Validate Password S: Allow to that Page or Access the Account.
Extension	2a 2b	Password is incorrect: S: Display the error message and retry for 2times. Password tried for 2 times still incorrect: S: Close that App.

Quiz

1. Which one of the following options is categorized as a black-box test technique?

a) A technique based on analysis of the architecture

b) A technique checking that the test object is working according to the detailed design

c) A technique based on the knowledge of past faults, or general knowledge of failures

d) A technique based on formal requirements

Quiz

2. An employee's bonus is to be calculated. It cannot be negative, but it can be calculated down to zero. The bonus is based on the length of employment:

- **less than or equal to 2 years – 0 bonus**
- **more than 2 years but less than 5 years – 5%**
- **5 to 10 years inclusively – 10%**
- **Longer than 10 years – 20%**

What is the minimum number of test cases required to cover all valid equivalence partitions for calculating the bonus?

a) 3

b) 5

c) 4

d) 2

Quiz

3. A fitness app measures the number of steps that are walked each day and provides feedback to encourage the user to keep fit. The feedback for different numbers of steps should be:

- **Up to 1000 - Couch Potato!**
- **Above 1000, up to 2000 - Lazy Bones!**
- **Above 2000, up to 4000 - Getting There!**
- **Above 4000, up to 6000 - Not Bad!**
- **Above 6000 - Way to Go!**

Which of the following sets of test inputs would achieve the BEST equivalence partition coverage?

a) 0, 1000, 2000, 3000, 4000

b) 1000, 2001, 4000, 4001, 6000

c) 123, 2345, 3456, 4567, 5678

d) 666, 999, 2222, 5555, 6666

Quiz

4. A daily radiation recorder for plants produces a sunshine score based on a combination of the number of hours a plant is exposed to the sun (below 3 hours, 3 to 6 hours or above 6 hours) and the average intensity of the sunshine (very low, low, medium).Given the following test cases:

	Hours	Intensity	Score
T1	1.5	v. low	10
T2	7.0	medium	60
T3	0.5	v. low	10

What is the minimum number of additional test cases that are needed to ensure full coverage of ALL VALID INPUT equivalence partitions?

a) 1

b) 2

c) 3

d) 4