



# **ISTQB Foundation Level – Syllabus V 4.0 2023**

## **Mohammed Sallam**

# CTFL-ISTQB Outline

Ch1-Fundamentals of Testing (8Q).

Ch2-Testing Throughout the Software Life Cycle (5Q).

Ch3-Static Techniques (5Q).

Ch4-Test Design Techniques (11Q).

Ch5-Test Management (9Q).

Ch6-Tools Support For Testing (2Q).

# Ch1 .Fundamentals of Testing

- What is The Testing?
- Why is Testing Necessary?
- Seven Testing Principles
- Fundamental Test Process – STLC vs. SDLC
- The Psychology of Testing –Deleted in v4

# What is Testing?

- Software testing is a **process of executing a program** to find software bugs in different level of SDLC (Requirement , Analysis , Design .. Etc.).
- Make sure everything's work as expected !
- To meets the business (Functional) and technical (Non Functional) requirements.
- Software that does not work correctly can lead to many problems, including:
  - 1.loss of money, time, or business reputation
  2. Injury or death (Embedded system).

# What is Testing?

- Software Testing  $\neq$  Test Execution
- Software testing is a process which includes many different activities
- Execution is only one of these activities.



# What is Testing?

Let's break down the basic definition of Software testing into the following parts:

1. Static Testing: find defects **without executing code**. For example: reviewing documents , walkthrough, inspection, etc.
2. Dynamic Testing: The software **code is executed** to Show the result of running tests.

# Why is Testing Necessary?

- Reduces the risk of problems occurring when users use the software finding software defects during software development to fix it before released.
- Ensure the quality of the product
- Makes sure that the customer satisfied

# Verification and Validation

- Verification : Are we building the **system right** ? We following process and guideline.
- Validation : Are we building the **right system**? As the client expected !



# Testing Objectives

| Need for Testing                    | Objective(s)  |
|-------------------------------------|---|
| Requirements Review                 | Preventing defects  |
| Development Testing                 | Finding defects<br>Reducing Risk  |
| Operational Testing- non-functional | Check system characteristics such as: Usability ,<br>availability , and security. |
| Acceptance Testing                  | Confirm the software works as expected.<br>Providing Information to Stakeholders  |
| Maintenance Testing                 | No new defects have been introduced during fixing<br>the changes.                 |

# Test Objectives (2)

- Evaluating work products such as requirements, user stories, designs, and code
- Triggering failures and finding defects
- Ensuring required coverage of a test object
- Reducing the level of risk of software quality
- Verifying whether specified requirements have been fulfilled
- Verifying that a test object complies with contractual, legal, and regulatory requirements
- Providing information to stakeholders to allow them to make informed decisions
- Building confidence in the quality of the test object
- Validating whether the test object is complete and works as expected by the stakeholders

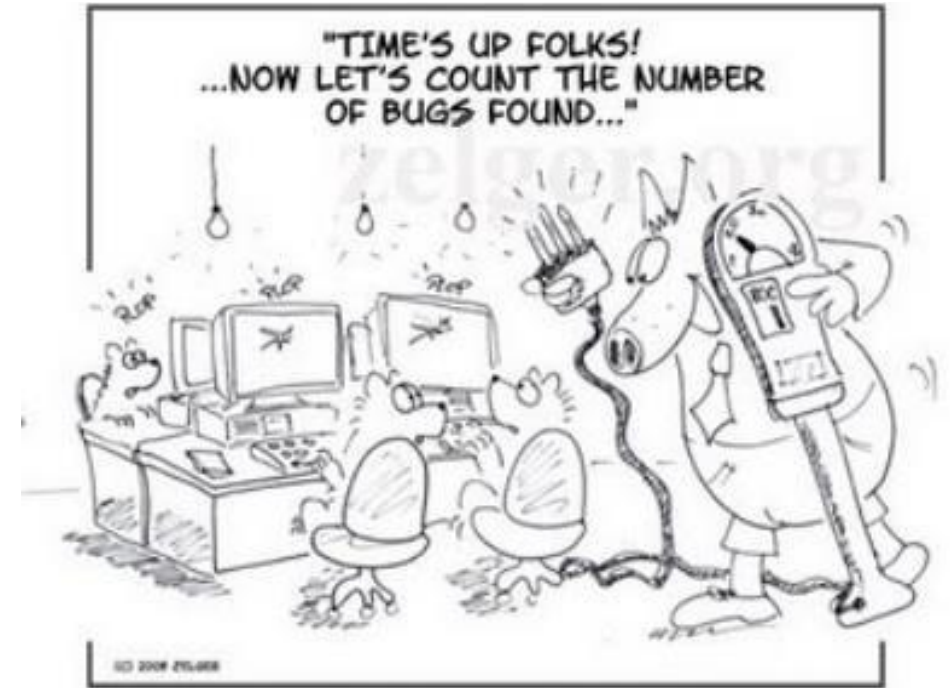
# Testing and Debugging

- Testing : Executing tests (test case) to show failures. Done by TESTER
- Debugging: is the development activity that finds, analyzes, and fixes issue. Done by DEVELOPER.

**NOTE:** In Agile development and in some other lifecycles, testers may be involved in debugging and component testing.

# How much Testing is enough?

- Prioritize tests so that, whenever you stop testing, **you have done the best testing in the time available.**
- What is the test suite that high priority ?



# Quality Management

- Includes all activities that direct and control an organization with regard to quality (QA and QC)



# QA vs. QC?

- **Quality assurance** focused on processes (Prevention Defect)
- **Quality control [Testing]** focused on product Quality (Detection Defect)



# Errors, Defects, and Failures

Error : human action, logic Error in code for example missing continue on while loop in JAVA.

Defect (bug): Any deviation of functional specification document ,When Actual result different of the expected result. For example as a user I want to login by email (Expected Result)  
But Actually system implemented to login by phone number .

Failures:Part of the system crashed.

Error

Defect

Failure



No failure



# Why do errors happen ?

- Time pressure
- Lack of Experience
- Miscommunication
- Complexity of work products
- System Interactions
- New technologies

# False Positive Vs. False Negative


## **False Positive [Invalid bugs] :**



- When a test case fails, but in actuality there is no bug and the functionality is working correctly.
- When execute test case using wrong test data will introduce defect.

## **False Negative :**

- Software already containing failure but not appears during testing.

# False positive or false negative ?

 **SOUQ**  
an amazon company


Log In  

**Labtob** (5 Items found)

Category | Brand | Price

☐ Hard Drives

Did you mean **Laptop**?



Seagate 1 TB Backup Plus  
USB 3.0 Slim Portable

**949.00 EGP**  
~~1,170.00 EGP~~

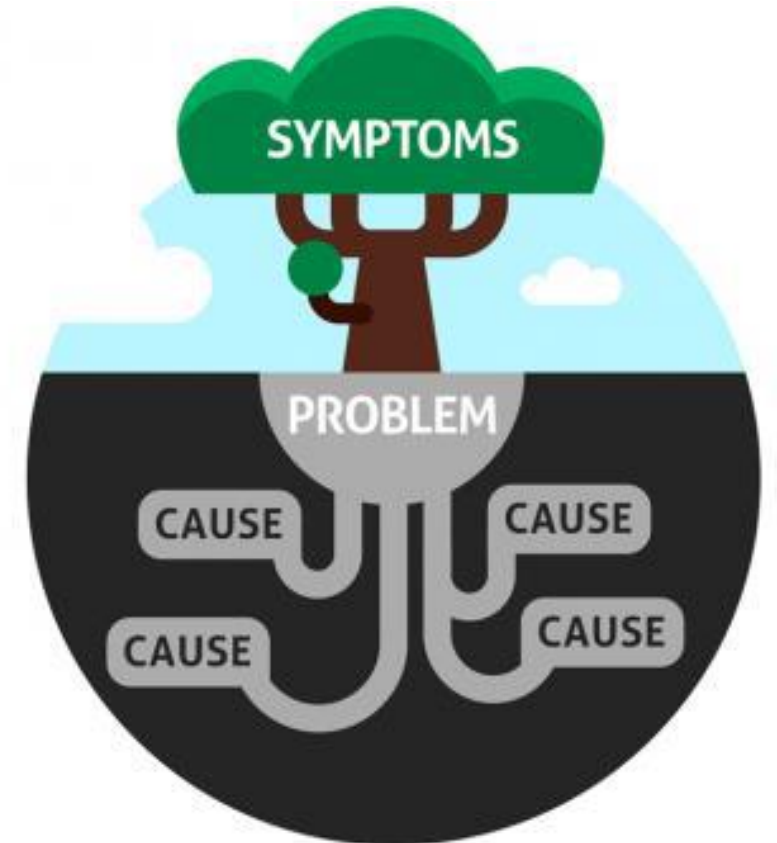
★★★★★

FREE Shipping

Fulfilled by souq

# Root Cause Analysis

The root causes of defects are the earliest actions or conditions that contributed to creating the defects.



**QUIZ  
TIME!**

Which of the following statements is a valid objective for testing?

A. To determine whether enough component tests were executed within system testing

**B. To find as many failures as possible so that defects can be identified and corrected**

C. To prove that all possible defects are identified

D. To prove that any remaining defects will not cause any failures

Which of the following statements correctly describes the difference between testing and debugging?

A. Testing identifies the source of defects; debugging analyzes the defects and proposes prevention activities

**B. Testing shows failures caused by defects; debugging finds, analyzes, and removes the causes of failures in the software**

C. Testing removes faults; debugging identifies the causes of failures

D. Testing prevents the causes of failures; debugging removes the failures

When the tester verifies the test basis while designing tests early in the lifecycle, which common test objective is being achieved?

- A. Gaining Confidence
- B. Finding Defects
- C. Preventing Defects**
- D. Providing information for decision making



Which of the following is an example of debugging?

- A. A tester finds a defect and reports it
- B. A tester retests a fix from the developer and finds a regression
- C. A developer finds and fixes a defect**
- D. A developer performs unit testing

Which of the following is the activity that removes the cause of a failure?

- A. Testing
- B. Dynamic Testing
- C. Debugging**
- D. Reverse Engineering

What type of activity is normally used to find and fix a defect in the code?

A. Regression Testing

**B. Debugging**

C. Dynamic analysis

D. Static analysis

Which one of the statements below describes a failure discovered during testing or in production?

- A. The product crashed when the user selected an option in a dialog box**
- B. The wrong version of one source code file was included in the build
- C. The computation algorithm used the wrong input variables
- D. The developer misinterpreted the requirement for the algorithm

Which of the following is a correct statement?

**A.A developer makes a mistake which causes a defect that may be seen as a failure during dynamic testing**

B.A developer makes an error which results in a failure that may be seen as a fault when the software is executed

C.A developer has introduced a failure which results in a defect that may be seen as a mistake during dynamic testing

D.A developer makes a mistake which causes a bug that may be seen as a defect when the software is executed

In what way does root cause analysis contribute to process improvement?

- A. Helps to better identify and correct the root cause of defects**
- B. Outlines how development teams can code faster
- C. Specifies the desired root causes to be achieved by other teams
- D. Contributes to the justification of future project funding

Which of the following is the correct statement?

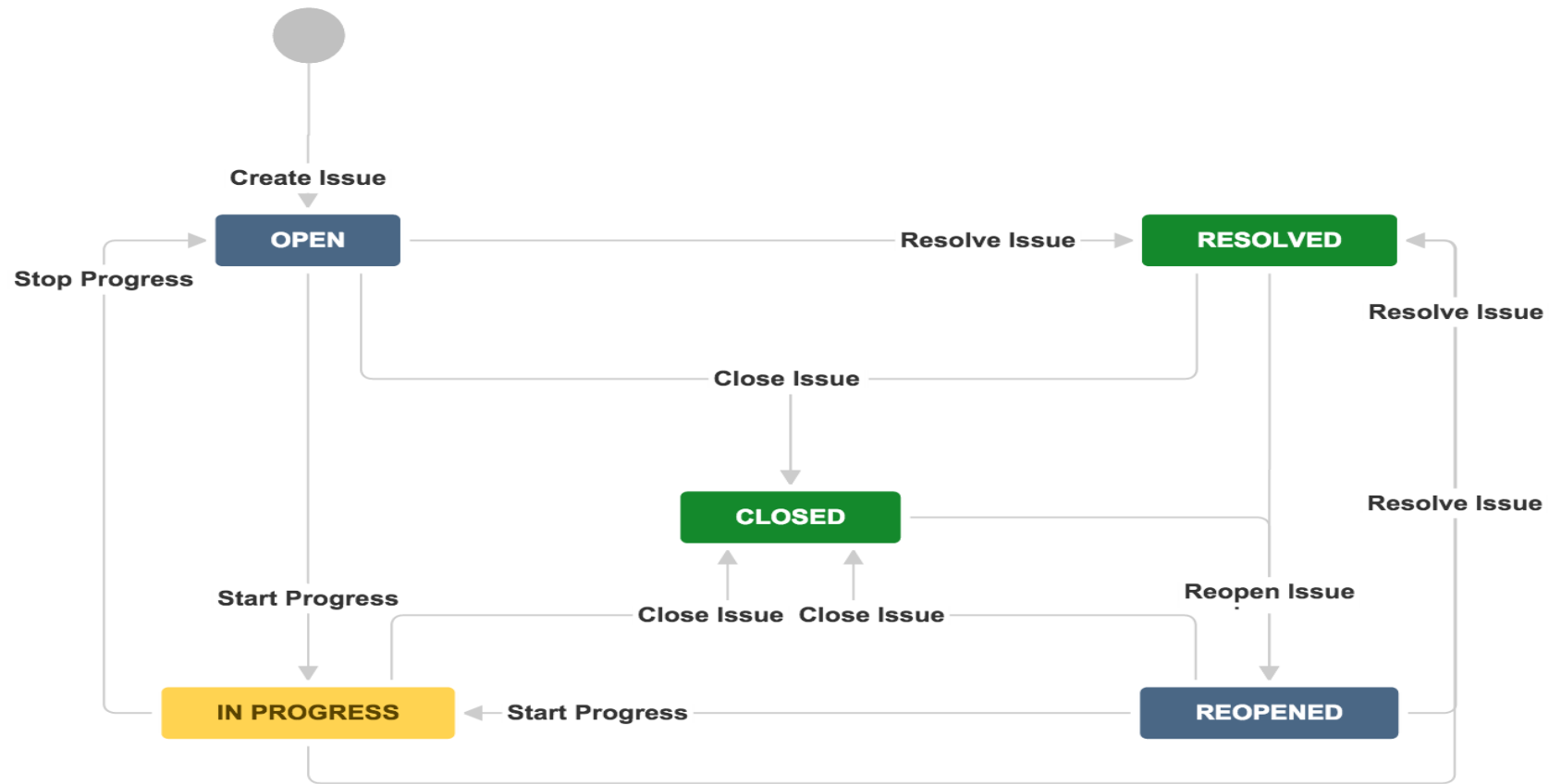
- A. An error causes a failure which results in a defect
- B. A defect causes a failure which results in an error
- C. A failure is observed as an error and the root cause is the defect
- D. An error causes a defect which is observed as a failure**

# Bug Report Contents

| Field               | Description   |
|---------------------|---|
| Summary             | A brief one-line summary of the bug   |
| <b>Descriptions</b> | The more detailed included <b>test step</b> , <b>actual result</b> and <b>expected result</b> to reproduce this bug |
| Report ID           | A unique identifier for this bug  |
| Project name        | The parent project to which the bug belongs.  |
| Priority vs. S      | how quickly the bug should be fixed and deployed<br>e.g., Low, Medium, and High                                     |
| Environment         | this issue occurred on any environment e.g. Production, pre-production, staging, and development.                   |
| Attachment          | Any screenshots, documents, voice notes, or video recordings that can assist in identifying and fixing the bug      |
| Assignee            | A person who is responsible to fix the bug. e.g. developer, Designer.   |
| Reporter            | A person who created bugs e.g. QA   |
| created date        | Bug submitted date.   |
| Status              | The stage the bug is currently at in its lifecycle (workflow).  |
| Fix version         | Project version(s) in which the bugs will be fixed.   |
| Component           | Bug component(s) to which this bug relates. e.g. Android, IOS, Backend (DB).  |



# Bug Report lifecycle -

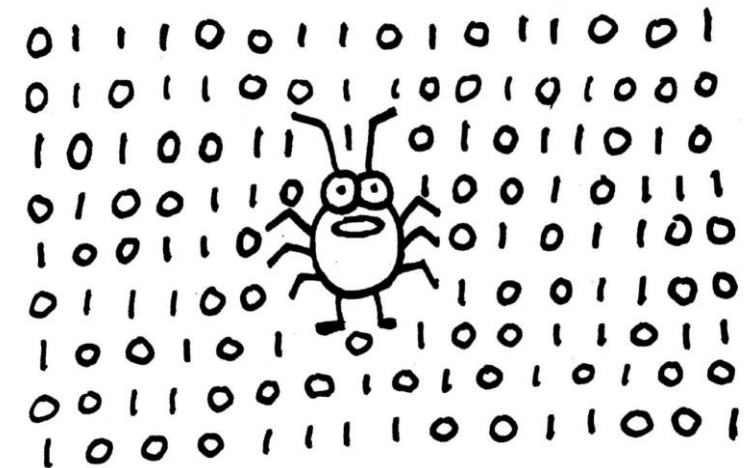


# Seven Testing Principles

1. Testing Shows Presence of Defects
2. Exhaustive Testing is impossible
3. Early Testing
4. Defect Clustering
5. Pesticide Paradox - Tests wear out
6. Testing is Context Dependent
7. Absence-of-Errors Fallacy

# 1) Testing Shows Presence of Defects

- Testing ... Prove that DEFECTS exist but cannot prove No DEFECTS exist.
- We cannot say that the product is **100% free defect**.
- Testing always reduces the number of undiscovered defect.
- $150 > 100 > 70 > 40$



## 2) Exhaustive Testing is impossible

- Testing everything including all combinations of inputs is not possible.
- We can use **priorities** to focus testing efforts.
- It is not feasible except in the case of trivial software



### 3) Early Testing

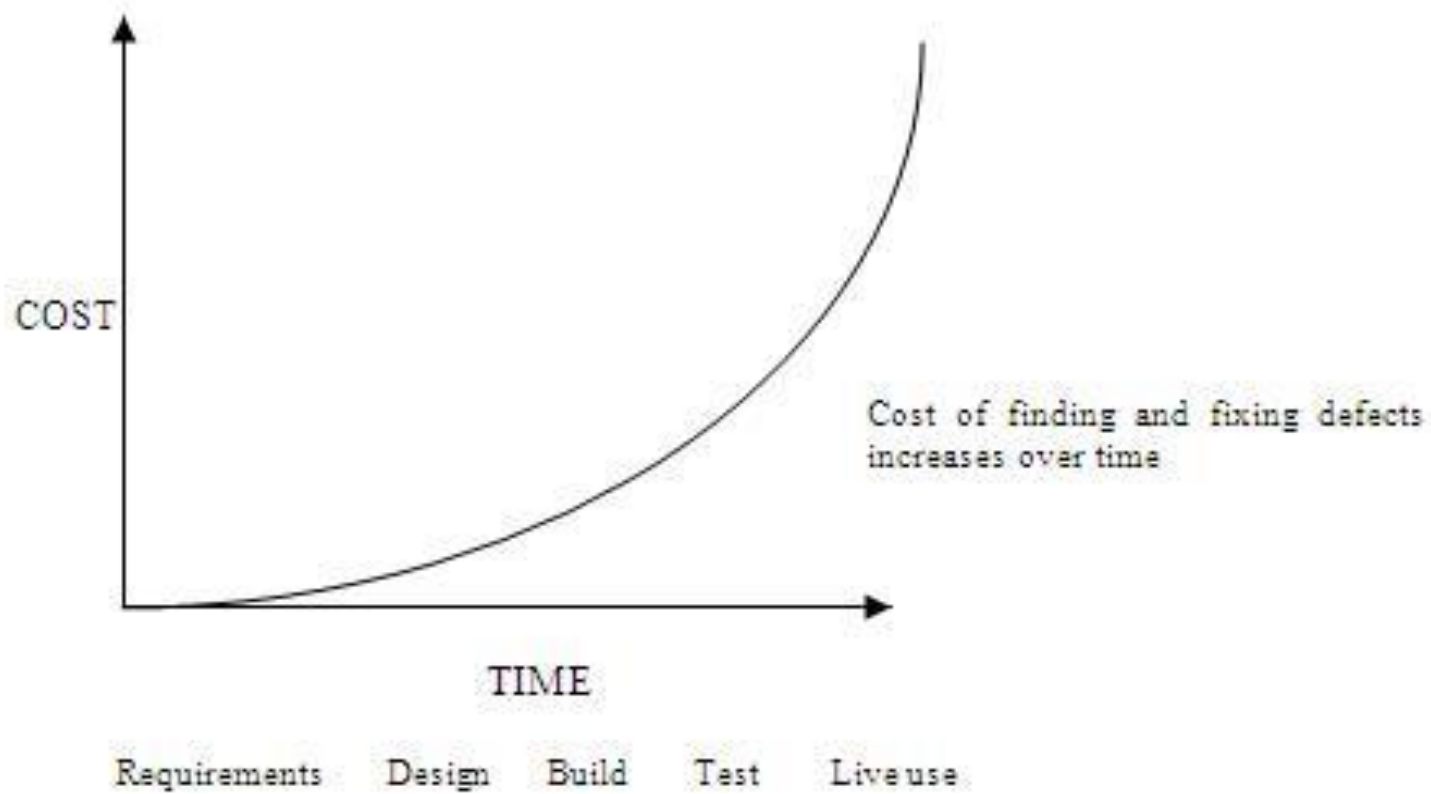
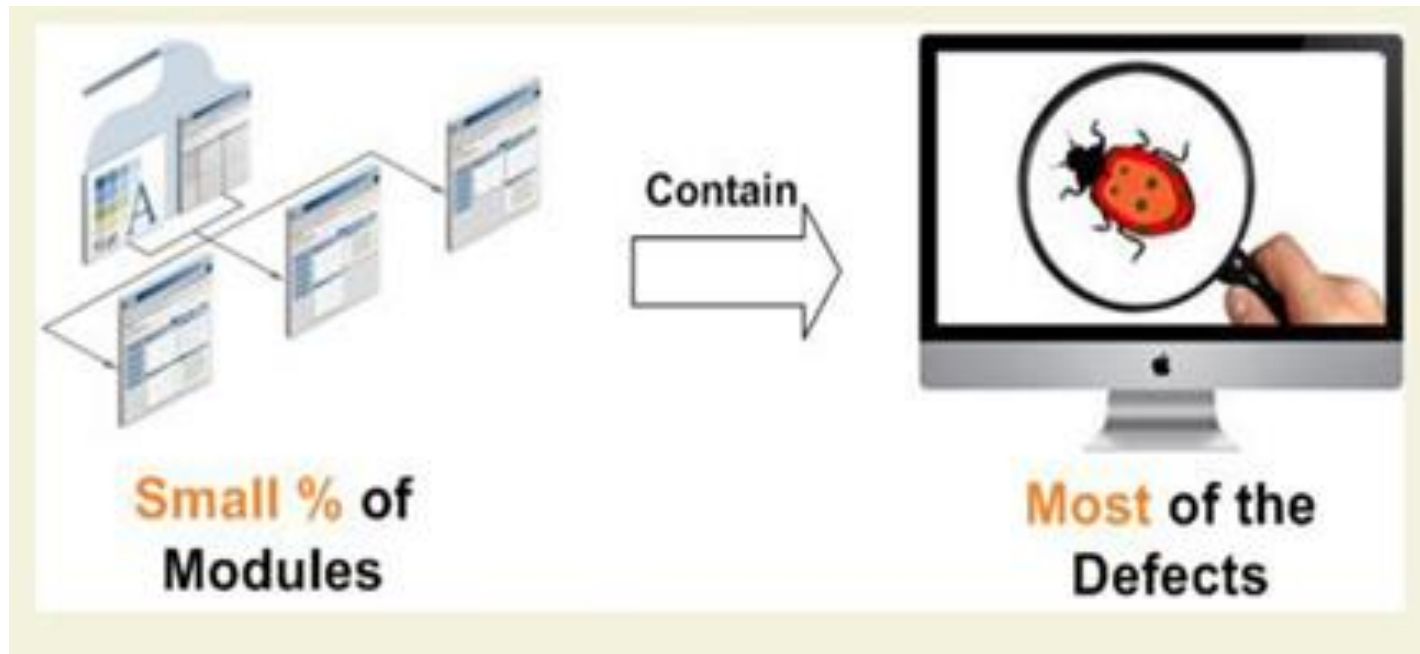


FIGURE 1.2

## 4) Defect Clustering

- A small number of modules contains most of the defects discovered.



## 5) Pesticide Paradox - Tests wear out

- If the same kinds of tests are repeated again and again with no longer be able to find any new bugs.
- Should review the test cases regularly , and add new test case.



## 6) Testing is Context Dependent

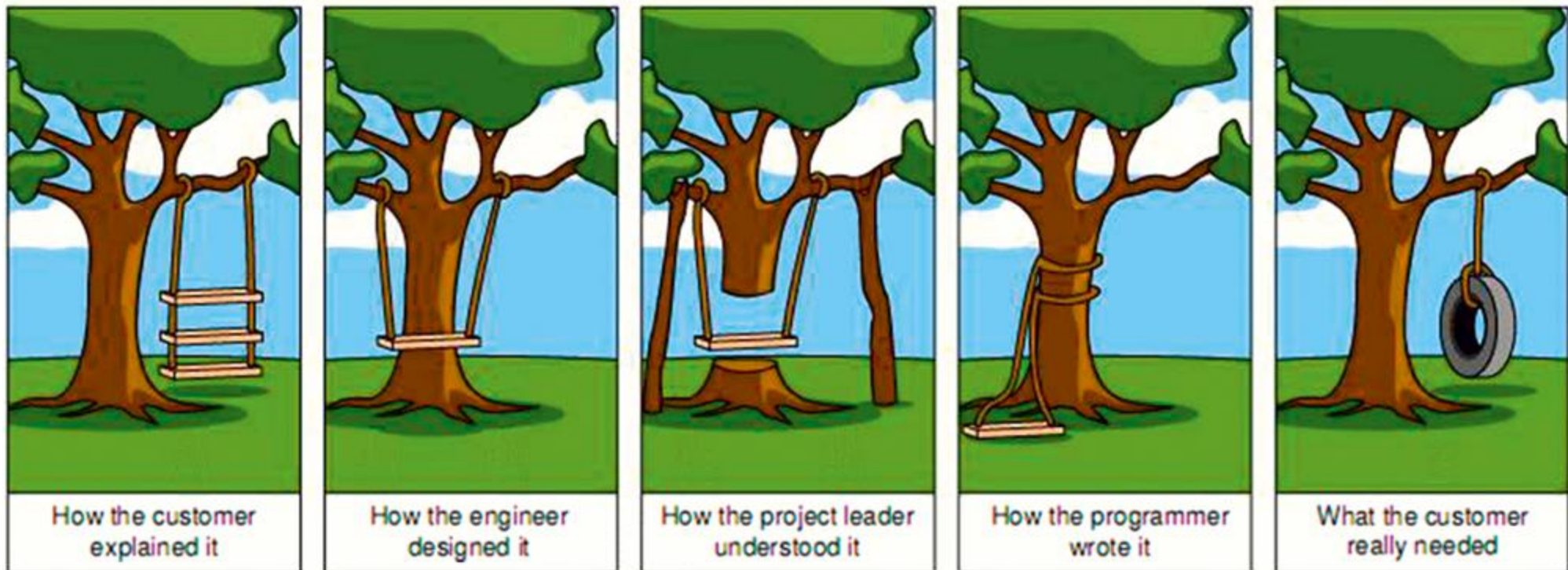
- Testing is basically context dependent Based on system , type of testing for e-commerce website is different of bank website.





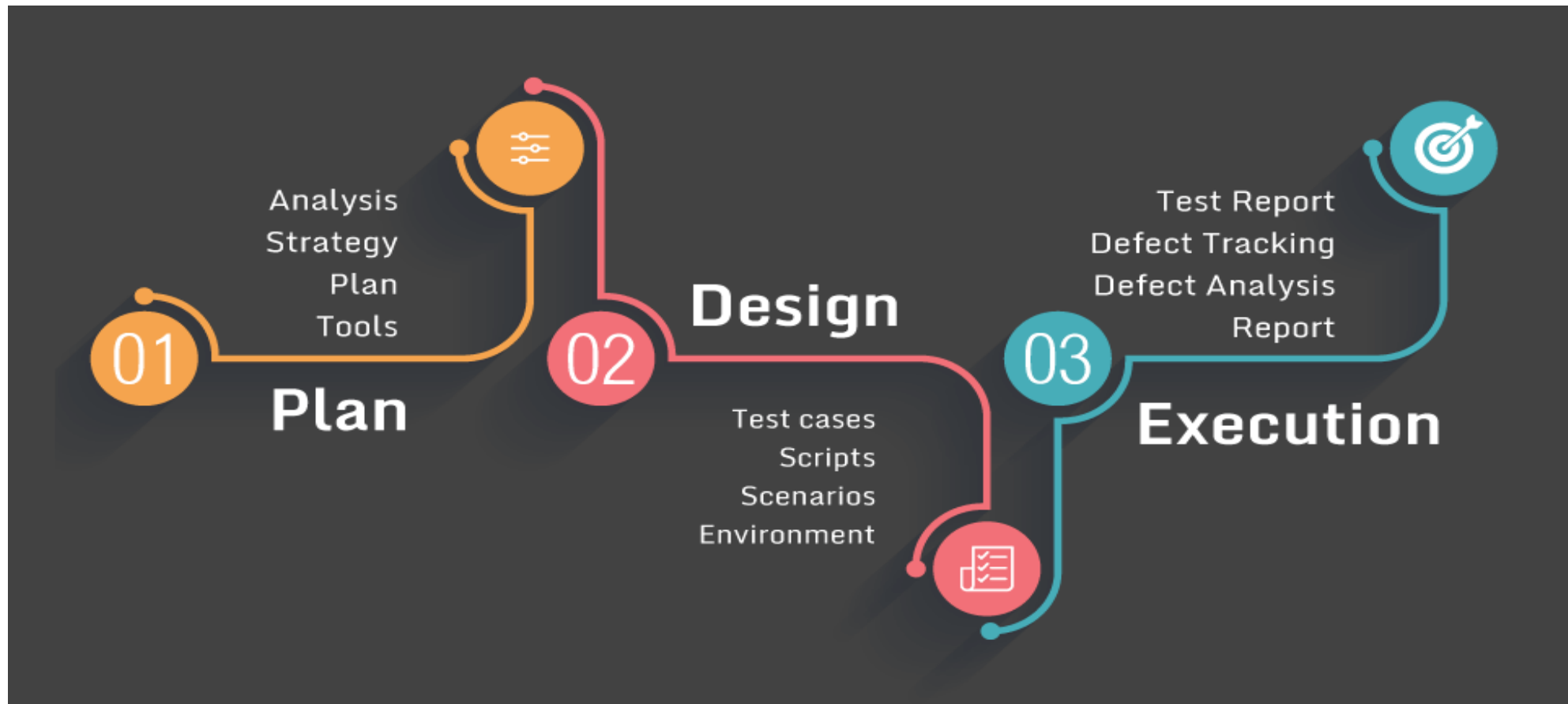
## 7) Absence-of-Errors Fallacy

- If the system built is unusable and does not fulfil the user's needs and expectations then finding and fixing defects does not help.
- Fulfill User needs ... Then ... Find Defects.



# Test Process - STLC vs. SDLC

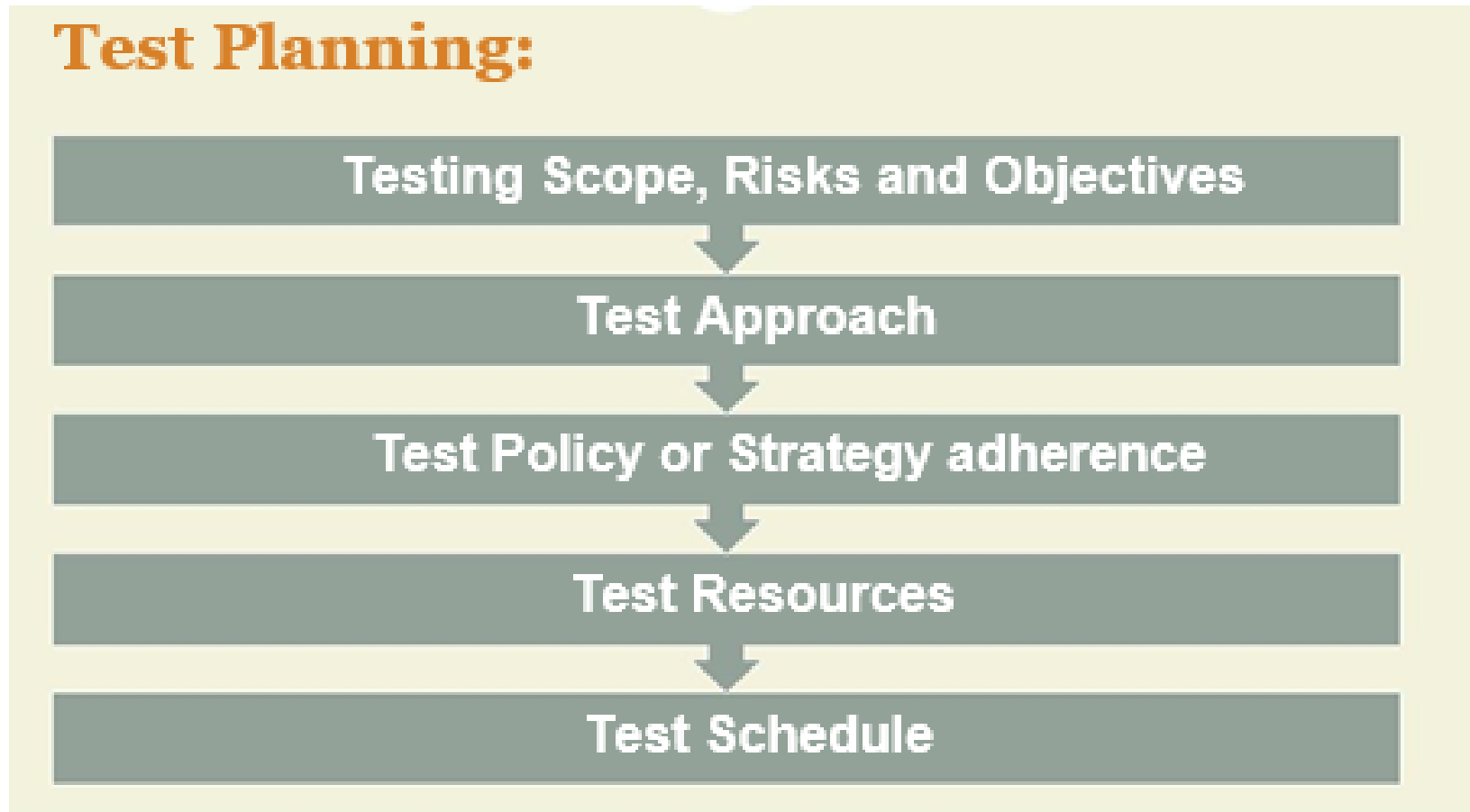
- There is no one **universal software test process**, but there are common sets of test activities.



# Test Activities



# Test Planning



# Test Monitoring & Progress + Control

**Test Control: Comparing actual progress against the plan & reporting the status**

**Measure & Analyze the results of testing**



**Monitor & Document progress of testing**



**Provide Regular testing status update**

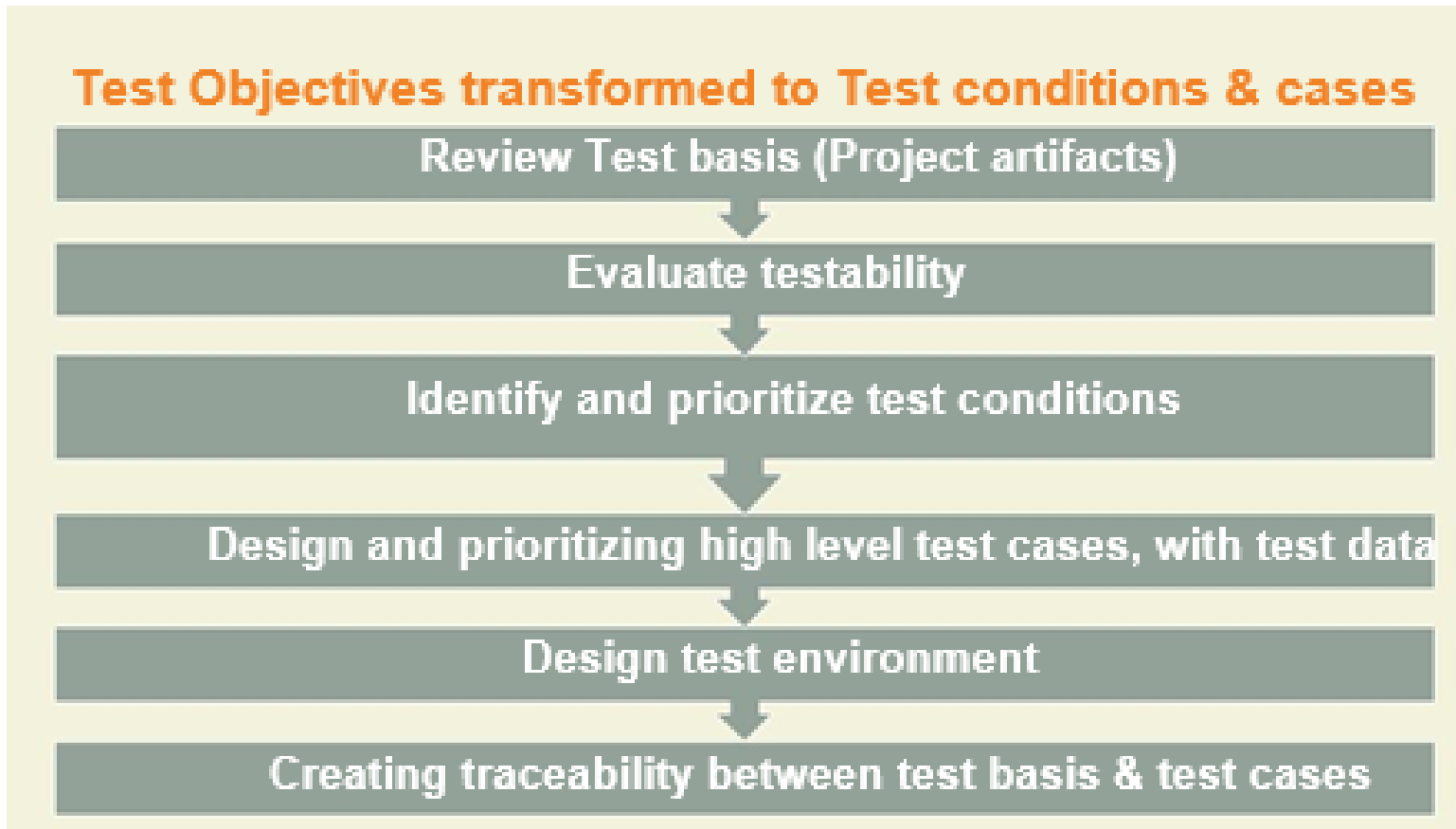


**Initiate corrective actions**



**Make decisions**

# Test Analysis ““ What To Test ?” and Test Design “How to Test”



# Test Implementation

**Test Implementation: Transform test conditions into test cases, procedures & data**

**And set up the test environment**

Finalizing, implementing and prioritizing the test cases



Developing & prioritizing test procedures, test data & scripts



Create test suites of test procedures



Verifying the correct setup of test environment



Verifying & updating traceability between test cases & test base

# Test Execution

**Test Execution: Executing the prepared test cases & report the results**

Execute test procedures & individual test cases

Log the outcome of test execution considering the version of SW

Compare Actual results with Expected results

Report discrepancies as incidents & investigate the results

Repeat test activities after fixing the discrepancies



# Evaluating Exit Criteria and Reporting - **Test Completion**

The activity where test execution is assessed against the defined objectives, to measure when “Enough” testing is achieved

Check the test log against the planned exit criteria



Assessing if more tests are needed or if the exit criteria should be changed



Writing a test summary report for stakeholders

# Test Closure Activities - Test Completion

**The activity of collecting data from completed test activities to consolidate experience & improve the process**

Check which planned deliverables we actually delivered



Closing incident reports/ raising change requests for deferred ones



Finalize and archive testware, environments & infrastructure



Handing over the testware to maintenance team



Analyzing lessons learned for future releases and projects

# Test Summary Report

|         |                |
|---------|----------------|
| PROJECT | HelpingTesters |
|---------|----------------|

|                                  |         |
|----------------------------------|---------|
| Overall progress of the QA cycle | On time |
|----------------------------------|---------|

|                            |     |
|----------------------------|-----|
| Total number of test cases | 100 |
|----------------------------|-----|

|                   |   |
|-------------------|---|
| Number of testers | 5 |
|-------------------|---|

|                     |        |
|---------------------|--------|
| Test cycle duration | 5 days |
|---------------------|--------|

|                       |
|-----------------------|
| Status for 10/27/2017 |
|-----------------------|

|                              |    |
|------------------------------|----|
| Number of test cases planned | 20 |
|------------------------------|----|

|                               |    |
|-------------------------------|----|
| Number of test cases executed | 18 |
|-------------------------------|----|

|                                       |    |
|---------------------------------------|----|
| Number of test cases executed overall | 78 |
|---------------------------------------|----|

|                                     |   |
|-------------------------------------|---|
| Number of defects encountered today | 2 |
|-------------------------------------|---|

|                                     |    |
|-------------------------------------|----|
| Number of defect encountered so far | 10 |
|-------------------------------------|----|

|  |   |
|--|---|
| Number of critical defects- still open | 3 |
|--|---|

|                |
|----------------|
| Overall status |
|----------------|

|                              |     |
|------------------------------|-----|
| Number of test cases planned | 100 |
|------------------------------|-----|

|                               |    |
|-------------------------------|----|
| Number of test cases executed | 78 |
|-------------------------------|----|

|                                |     |
|--------------------------------|-----|
| Pass Percentage of the defects | 98% |
|--------------------------------|-----|

|                 |             |
|-----------------|-------------|
| Defects density | 2.5 per day |
|-----------------|-------------|

|                             |     |
|-----------------------------|-----|
| Critical defects percentage | 20% |
|-----------------------------|-----|

# Testware - Test Work Products

- Testware is created as **output** work products from the test activities.

| Test Activities             | Work Products   |
|-----------------------------|---|
| Test planning               | Test plan, test schedule, Risk , entry and exit criteria.   |
| Test monitoring and control | Test progress reports   |
| Test analysis               | Test conditions (e.g., acceptance criteria)   |
| Test design                 | Test cases, test charters, coverage items, test data requirements and test environment requirements.  |
| Test implementation         | Test procedures, automated test scripts, test suites, test data, test execution schedule, and test environment elements.  |
| Test execution              | Test logs, and defect reports   |
| Test completion             | Test completion report (Test Summary Report ), action items for improvement of subsequent projects or iterations, documented lessons learned, and change requests (e.g., as product backlog items). |

# Traceability between the Test Basis and Testware

- Traceability of test cases to requirements can verify that the requirements are covered by test cases
- Traceability of **test results** to **risks** can be used to evaluate the level of remaining risk in a test object.

# Test Leader and Tester - Roles

Test Leader & Testers are the Main Two roles in Testing Team



# Test Leader's Tasks (Test management)

- Test policy-Test Strategy-Test Plan
- Test monitoring & Control (Test progress report-test summary report)
- Initiate the analysis, design, implementation, and execution of tests
- Configuration Management (CM)
- Metrics
- Tools selection
- Test Environment Implementation Decision
- Develop the skills and careers of testers



# The Tester

A skilled professional who is involved in the testing of a component or system.

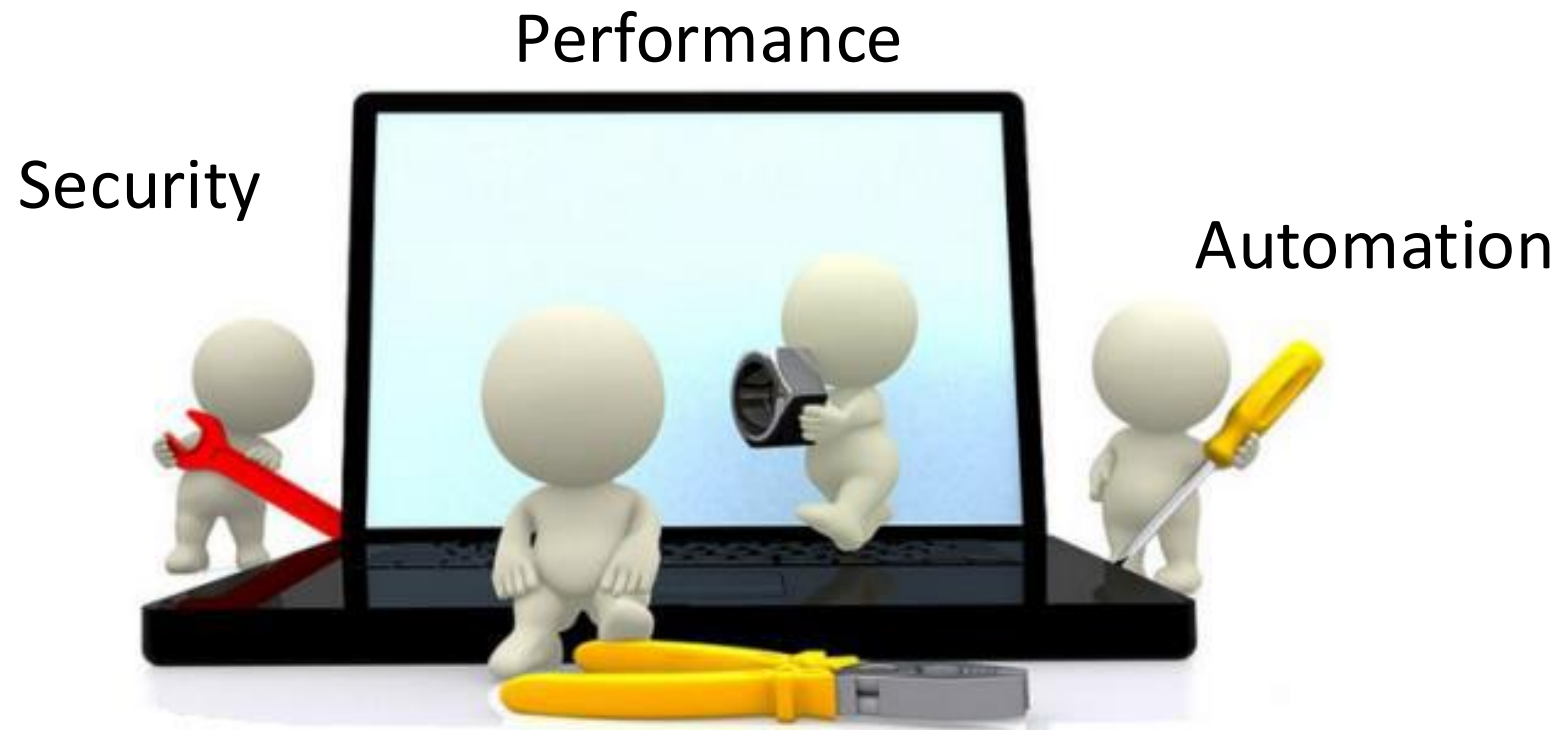




# Tester's Tasks

- Review and contribute to test plans
- Assess requirements for testability
- Test conditions, test cases, test procedures, test data, & test execution schedule
- Test Environment setup
- Test Execution
- Test automation
- Non-Functional Testing
- Review tests developed by others

# Specialized Testers (out)



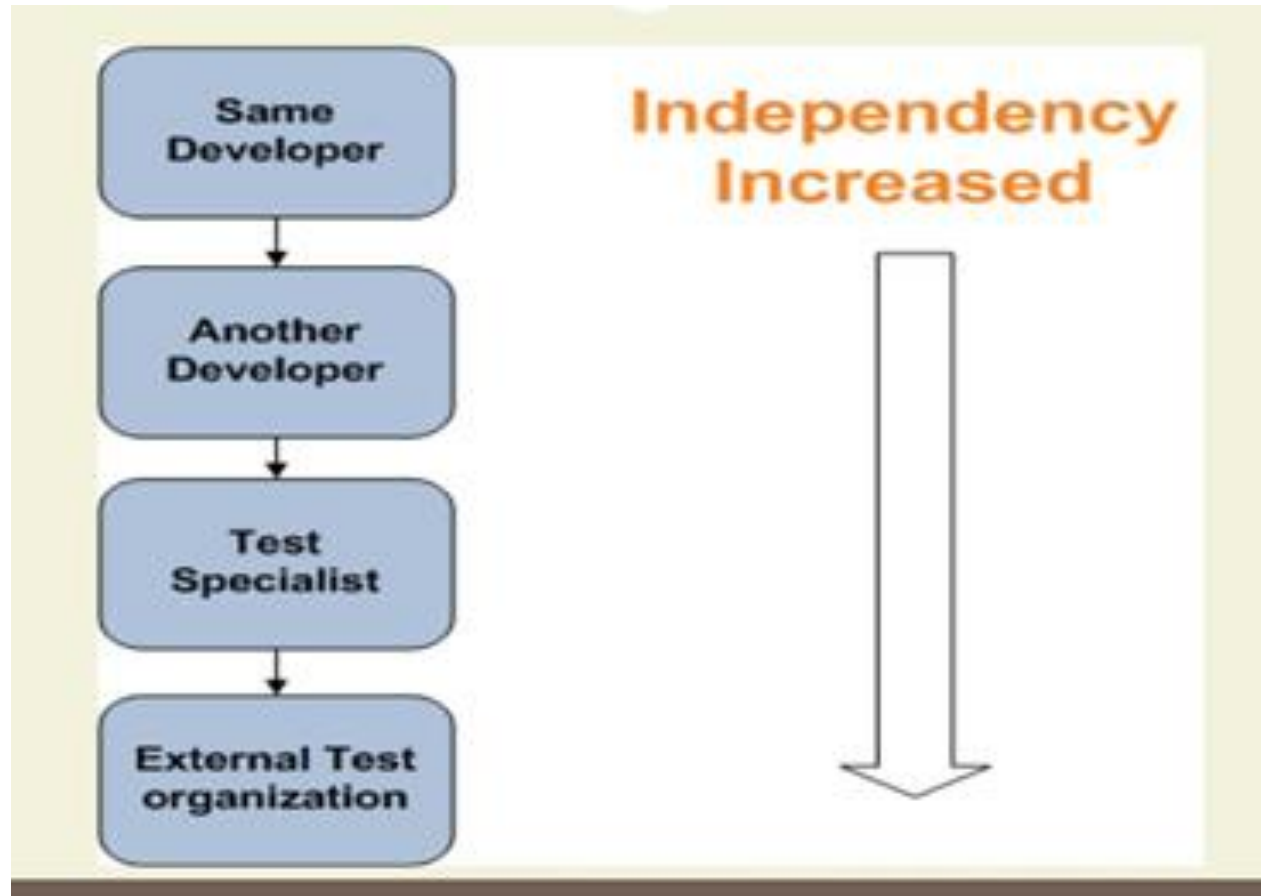
# Want to be a good Tester? (OUT)



# Whole Team Approach

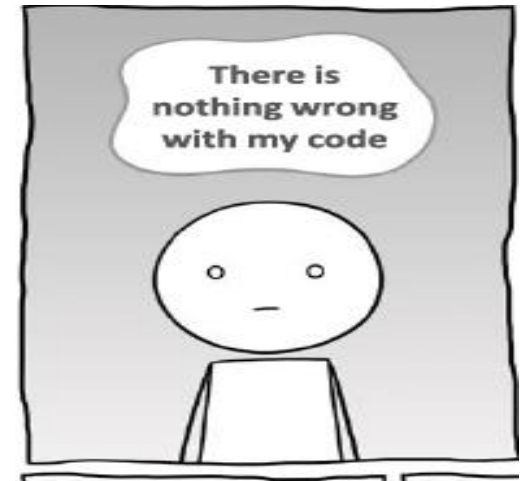
- Team member with the necessary knowledge and skills can perform any task, and **everyone is responsible for quality.**
- Cross Functional team
- The team members share the same **workspace** (physical or virtual), as co-location facilitates **communication** and **interaction.**
- Testers work closely with other team members to ensure that the quality levels are achieved.

# Level of Independence



# What is the Psychology of testing? (out)

- There should be clear communication and feedback on defects between tester and developer.
- Tester when reporting bugs should focus on system behavior not on person who created the bug!
- Since developers expect their code to be correct, difficult to accept that the code is incorrect.



**QUIZ  
TIME!**

Which of the following statements CORRECTLY describes one of the seven key principles of software testing?

A. By using automated testing it is possible to test everything

B. With sufficient effort and tool support, exhaustive testing is feasible for all software

**C. It is impossible to test all input and precondition combinations in a system**

D. The purpose of testing is to prove the absence of defects



Which of the following is a true statement about exhaustive testing?

- A. It is a form of stress testing
- B. It is not feasible except in the case of trivial software**
- C. It is commonly done with test automation
- D. It is normally the responsibility of the developer during unit testing

Why is it important to avoid the pesticide paradox?

- A. Dynamic testing is less reliable in finding bugs
- B. Pesticides mixed with static testing can allow bugs to escape detection
- C. Tests should not be context dependent
- D. Running the same tests over and over will reduce the chance of finding new defects**

Which of the below tasks is performed during the test analysis activity of the test process?

- A. Identifying any required infrastructure and tools
- B. Creating test suites from test scripts
- C. Analyzing lessons learned for process improvement
- D. Evaluating the test basis for testability**

The following test work products,1-4, by mapping them to the right description, A-D.

1.Test suite

2. Test case

3. Test script

4. Test charter

A.A group of test scripts or test execution schedule

B.A set of instructions for the automated execution of test procedures

C. Contains expected results

D. An event that could be verified

**a)1A, 2C, 3B, 4D.**

b)1D, 2B, 3A, 4C.

c)1A, 2C, 3D, 4B.

d)1D, 2C, 3B, 4A.

When following the fundamental test process, when should the test control activity take place?

- A. During the planning activities
- B. During the implementation and execution activities
- C. During the monitoring activities
- D. During all the activities**

If you need to provide a report showing test case execution coverage of the requirements, what do you need to track?

**A. Traceability between the test cases and the requirements**

B. Coverage of the risk items by test case

C. Traceability between the requirements and the risk items

D. Coverage of the requirements by the test cases that have been designed

Which of the following is the activity that compares the planned test progress to the actual test progress?

**A. Test monitoring**

B. Test planning

C. Test closure

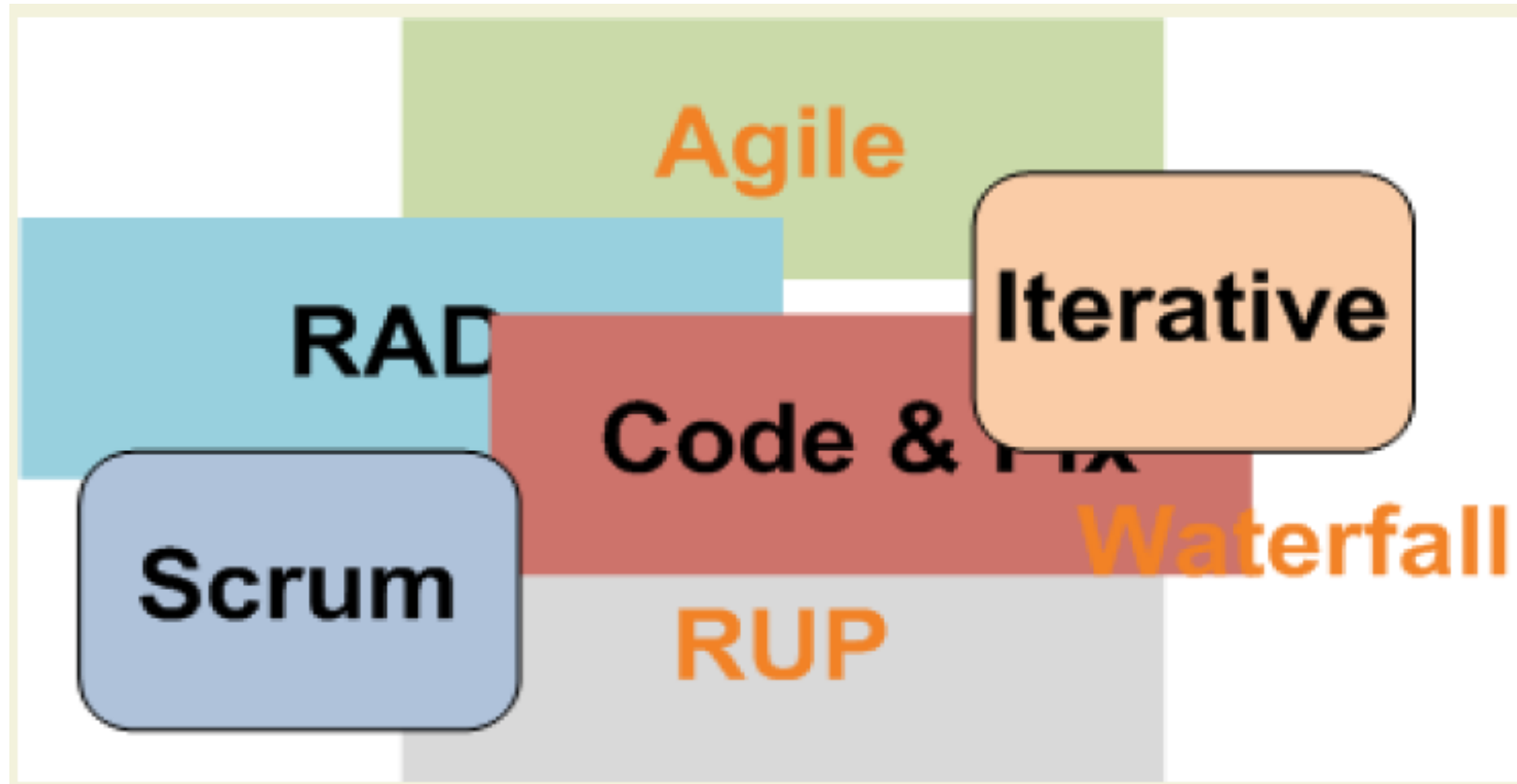
D. Test control

# Chapter 2: Testing Throughout the Software Lifecycle - SDLC

- Software Development Models SDLC
- Test Levels
- Test Types
- Maintenance Testing
- Testing as a Driven for Software Development ( TDD , ATDD , BDD)



# Software Development Models



**A software development lifecycle model** describes the types of activity performed at each stage in a software development project.

# Characteristics of good Testing In any software lifecycle model

- For every development activity, there is a corresponding test activity
- Testers participate in the discussions to define requirements , design, and are involved in reviewing work products.

# Software Development Models



**Sequential Model**

**Iterative-Incremental  
Model**



# Sequential Model

- Describes the software development process as a linear, sequential flow of activities.  
A>B>C
- This means that any phase in the development process should be start when the previous phase is complete.
- no overlap of phases

# Which Model to select?



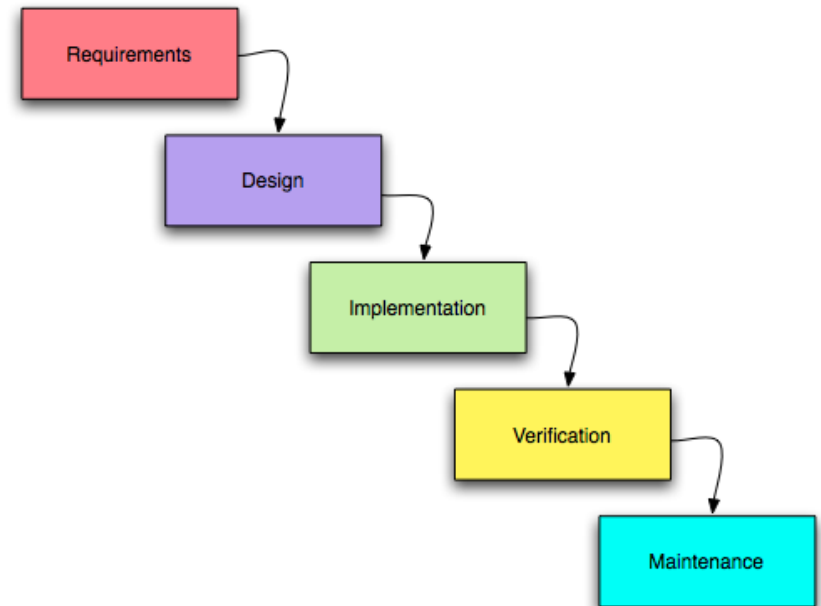
**Project Goals / Objectives**



**Project Risks**

# 1) Waterfall Model

- In the Waterfall model, the development activities are completed one after another.
- In this model, **test activities** only occur after all other development activities have been completed



# Advantage & disadvantage Waterfall Model

- **Advantage**

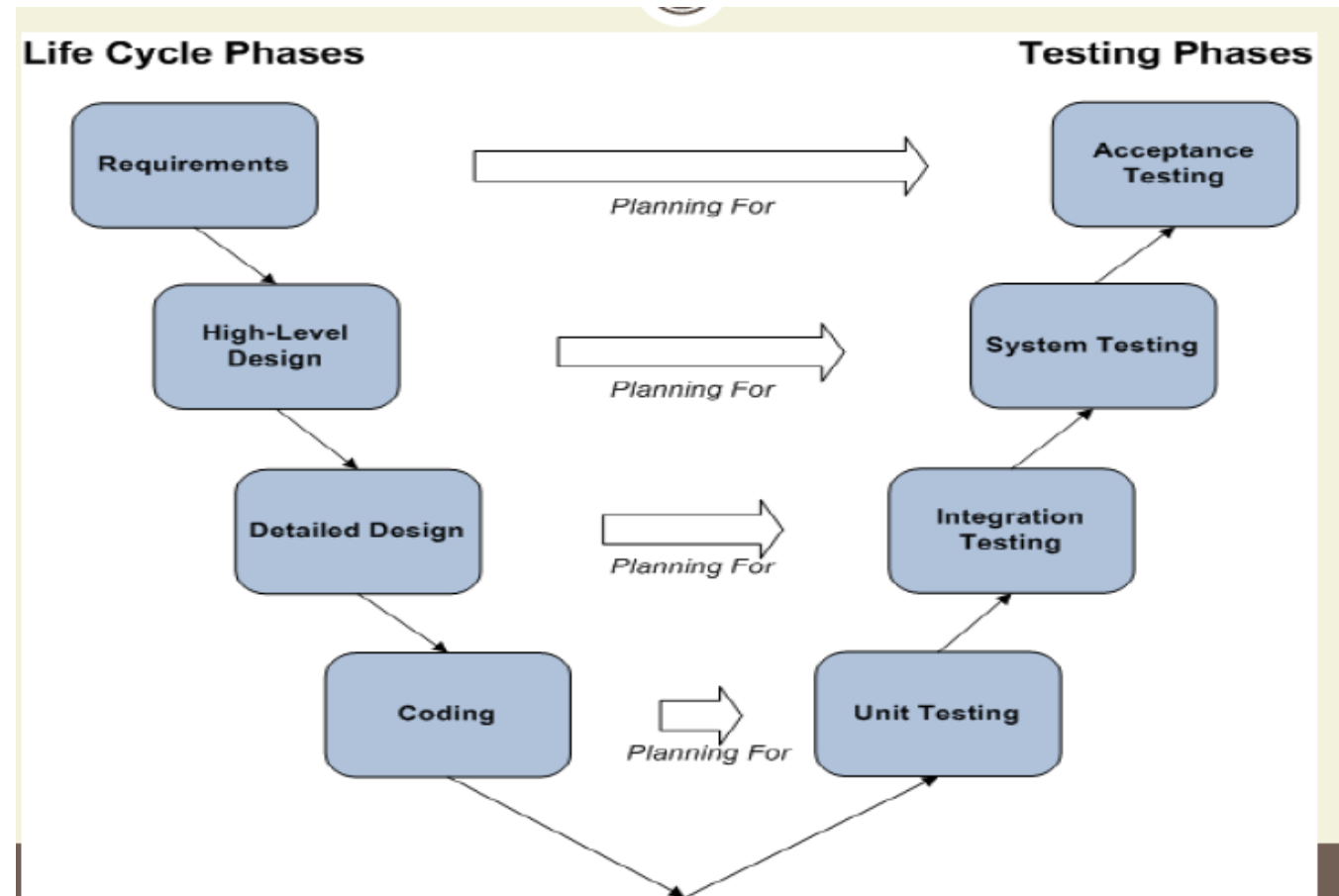
- 1- This model is simple and easy to understand and use.
- 2- Phases do not overlap.
- 3- **Waterfall model** works well for smaller projects where requirements are clearly defined and very well understood.

- **Disadvantage**

- 1- Once an application is in the testing stage, it is very difficult to go back and change something.
- 2- No working software is produced until late during the life cycle.
- 3- Not a good model for complex.

## 2) V-Model

- V-model integrates the test process throughout the development process, implementing the principle of early testing.





# Incremental & Iterative Development Model

- **Incremental development** involves establishing requirements, designing, building, and testing a system in pieces, which means that the software's features grow incrementally. **but typically require months or years for delivery to stakeholders and users.**
- **Iterative (Sprint )Development** occurs when groups of features are specified, designed, built, and tested together in a series of cycles, often of a **fixed duration**, iteration delivers working software which is a growing subset of the overall set of features.

Project : Shopping Project e-c  
Epic : Home page , Login , reg , Cart, my account  
Story

Login by phone number

Task IOS

Task Android

Task Web

Task BE [DB + API]

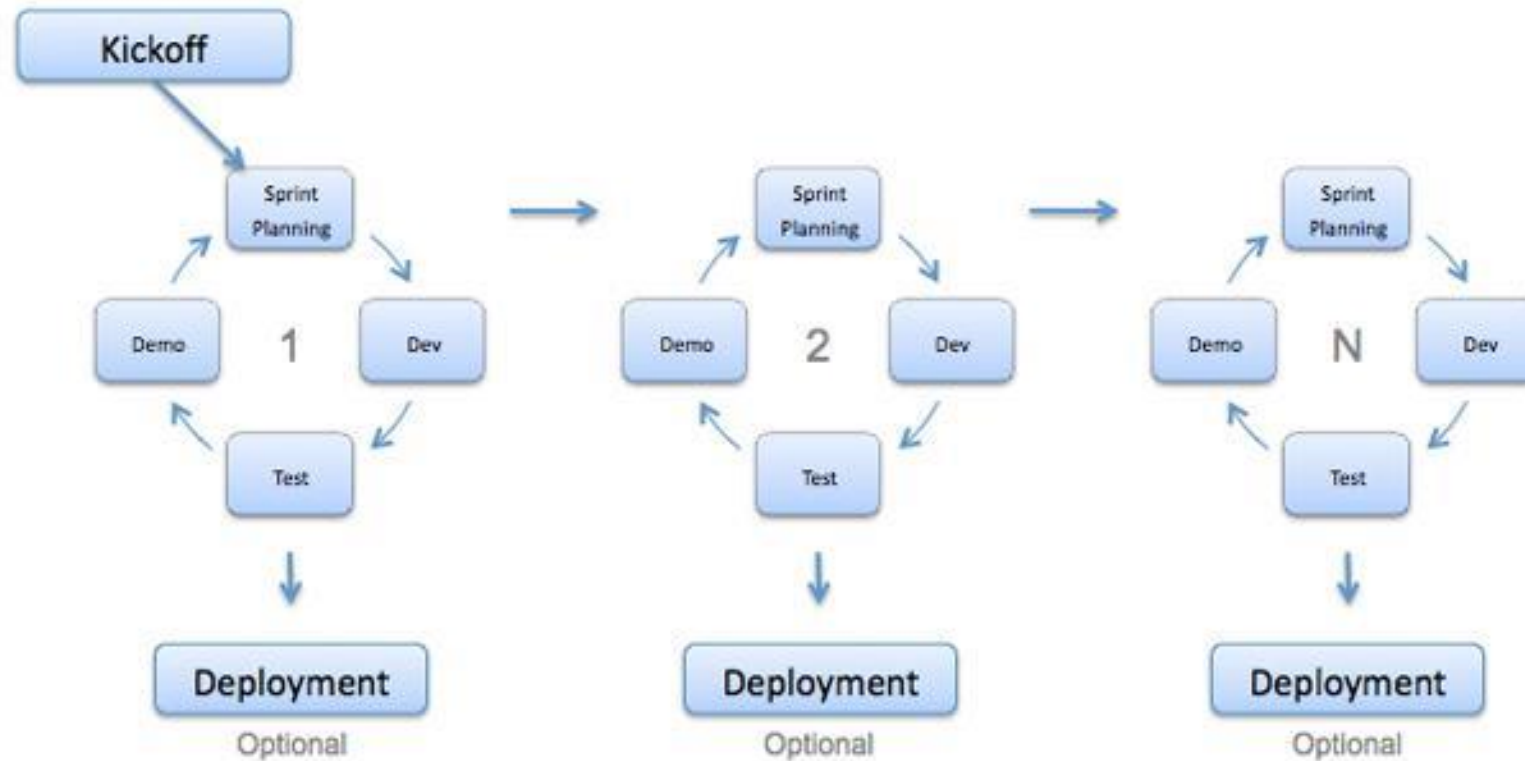
Task Testing

Login by gmail

Login by FB

Task , Test Case -> Bug report

### 3) Agile model



# Advantage & disadvantage Agile Model

- **Advantage**

- 1- Customer satisfaction by continuous delivery CD of useful software.
- 2- Face-to-face conversation is the best form of communication by daily standup meeting.
- 3- Even late changes in requirements are welcomed
- 4- Working software is delivered frequently (2-4 weeks).

- **Disadvantage**

- 1-Difficult to estimate the effort required at the beginning of the software development life cycle.
- 2- lack to focus on designing and documentation.

## 4)RAD model

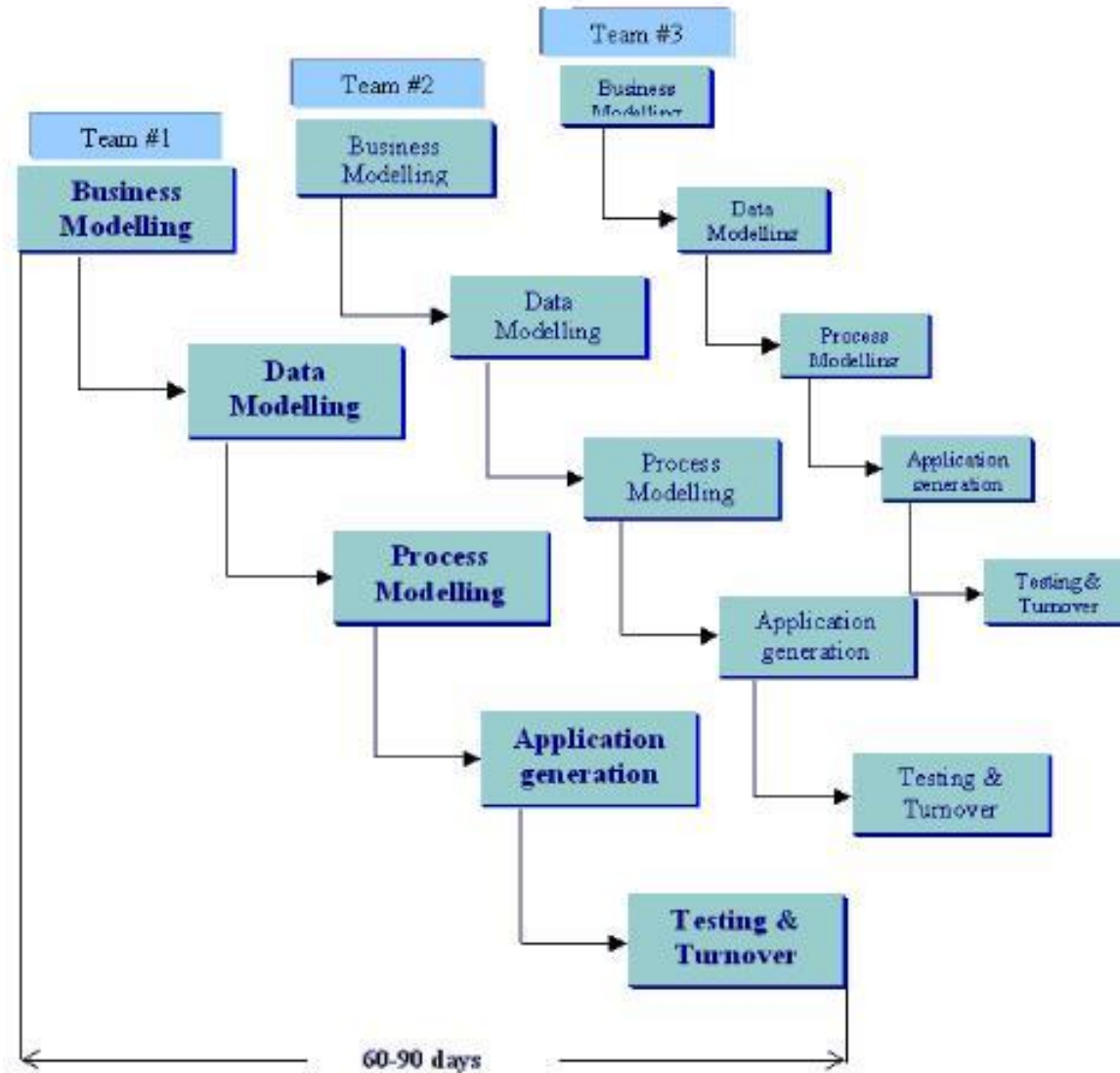
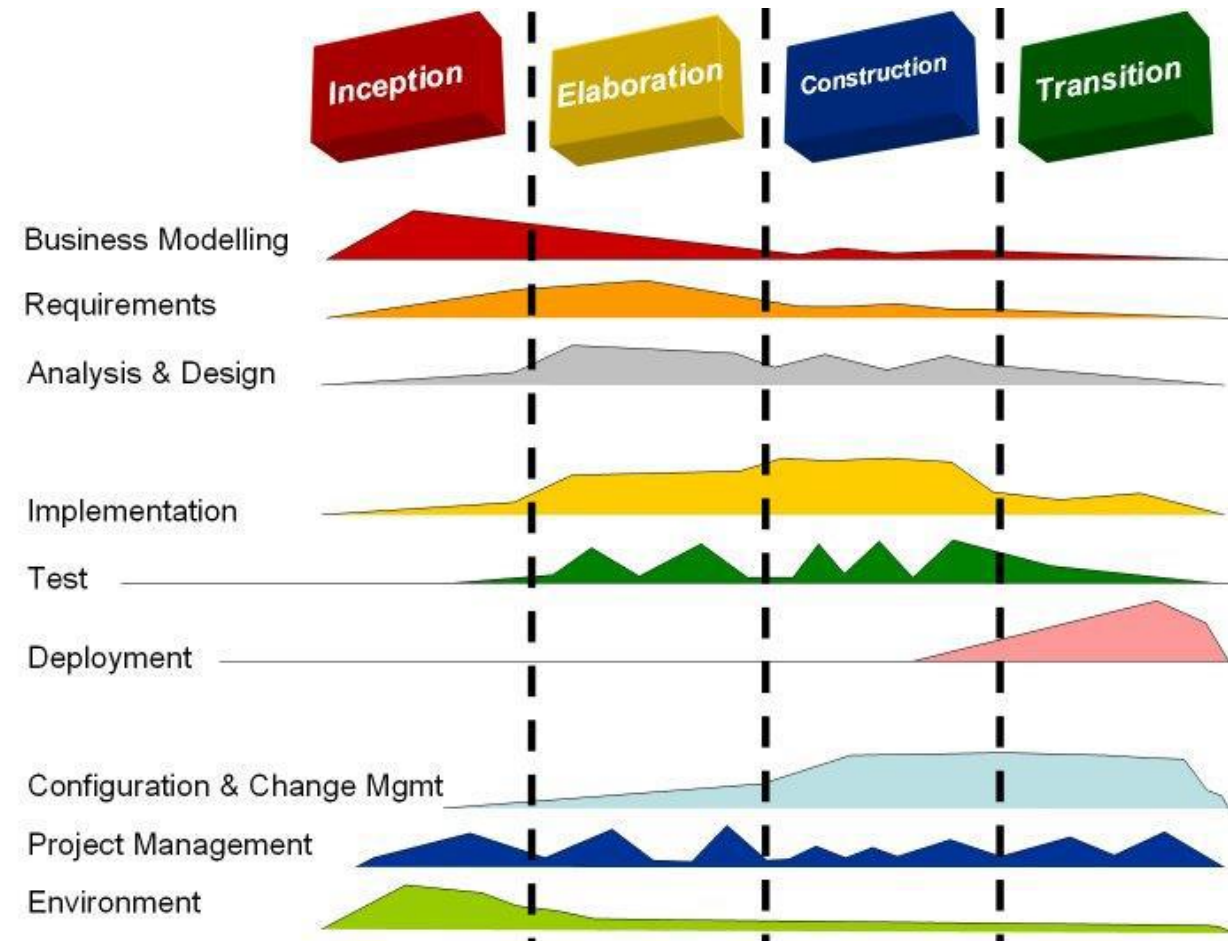


Figure 1.5 – RAD Model

# 5) Rational Unified Process

- Each iteration tends to be **relatively long** (e.g., two to three months), and the feature increments are correspondingly large, such as two or three groups of related features. -> Epic



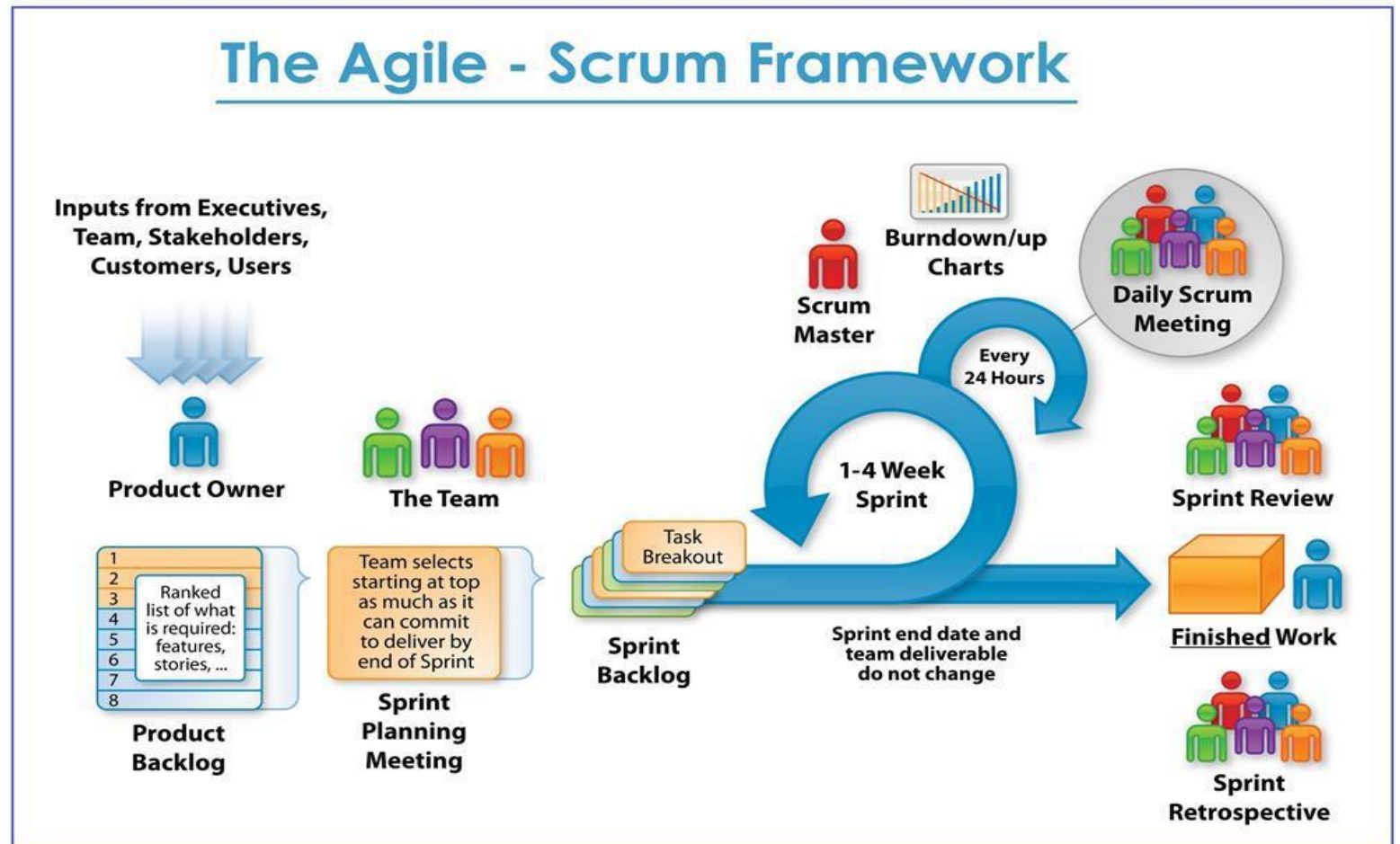
# THREE MAIN SCRUM ROLES





## 6) Scrum

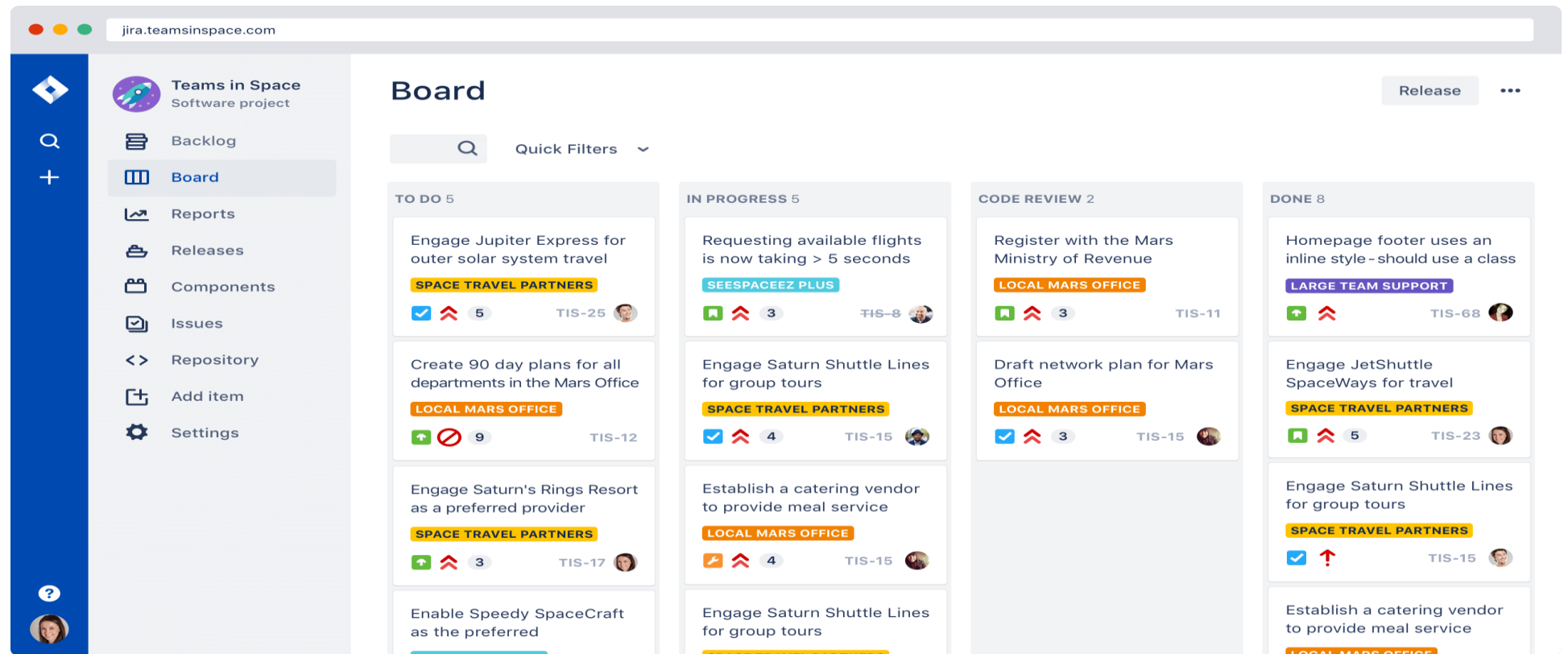
Each iteration tends to be **relatively short** (e.g., hours, days, or a few weeks), and the feature increments are correspondingly small, such as a few enhancements and/or two or three new features.





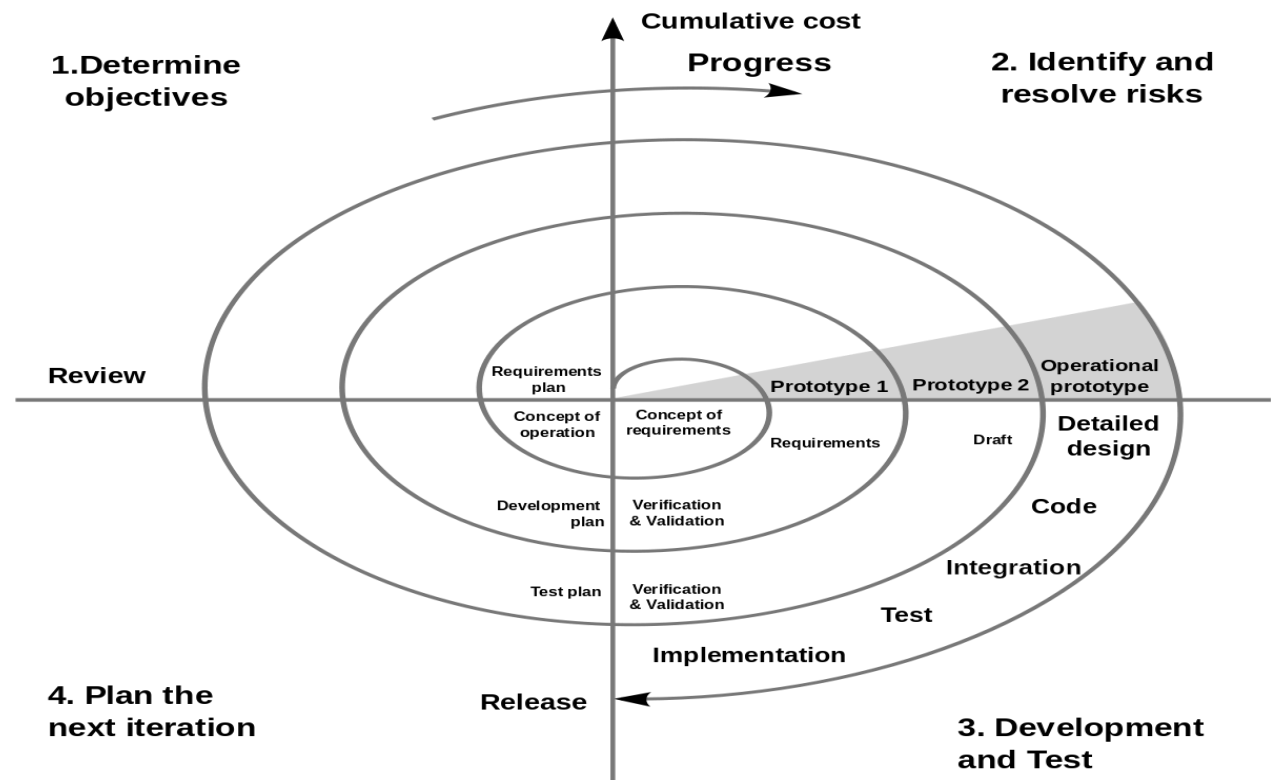
# 7) Kanban

Implemented **with or without fixed-length iterations**, which can deliver either a single enhancement or feature upon completion, or can group features together to release at once.



# 8- Spiral (or Prototyping)

- Involves creating experimental increments, some of which may be heavily re-worked or even abandoned in subsequent development work.



**QUIZ  
TIME!**

Which one of the following is the BEST definition of an incremental development model?

**A. Defining requirements, designing software and testing are done in a series with added pieces**

B. A phase in the development process should begin when the previous phase is complete

C. Testing is viewed as a separate phase which takes place after development has been completed

D. Testing is added to development as an increment

Which of the following is a true statement regarding the V-model lifecycle?

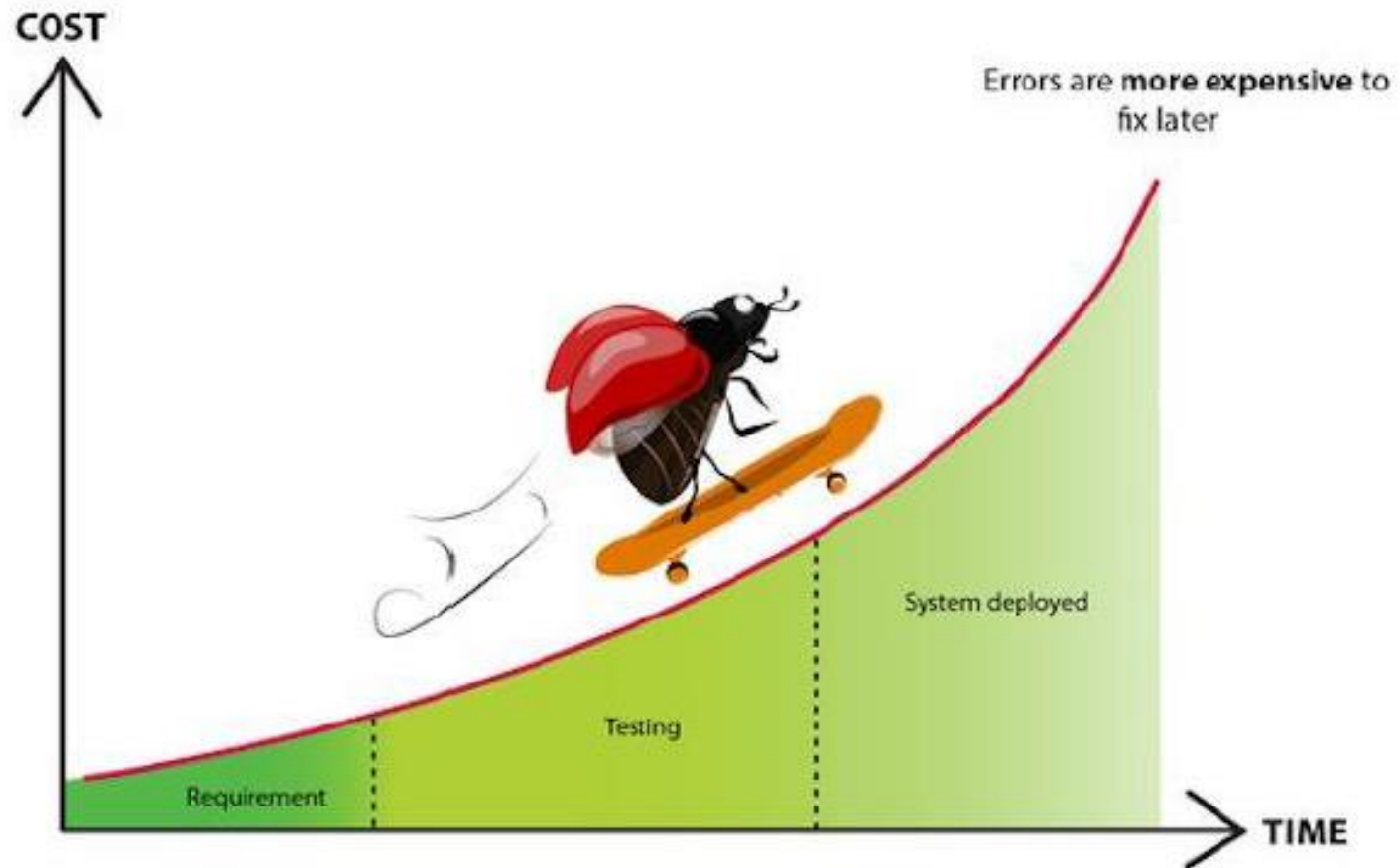
A. Testing involvement starts when the code is complete

**B. The test process is integrated with the development process**

C. The software is built in increments and each increment has activities for requirements, design, build and test

D. All activities for development and test are completed sequentially

# Cost of fixing faults

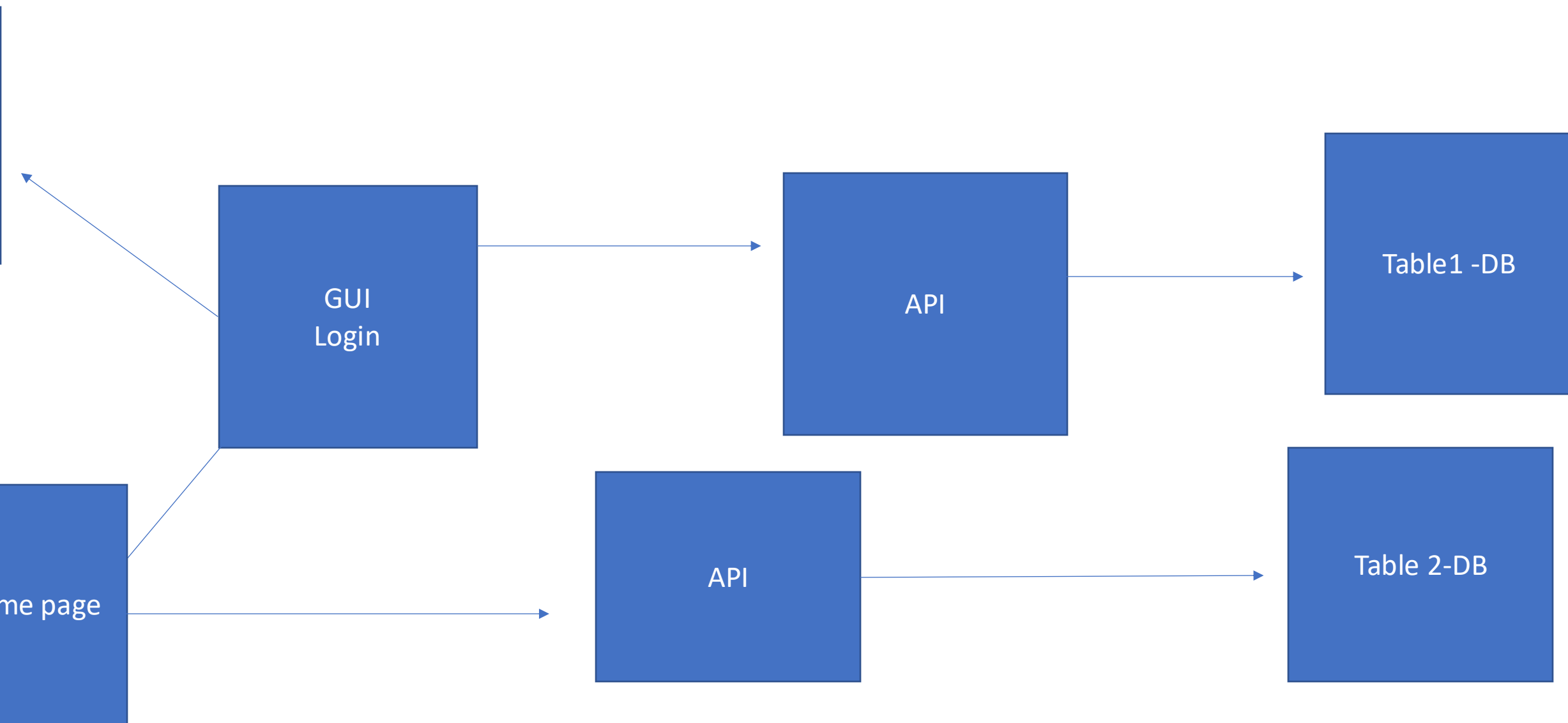


# Test Levels

## Test Level:

A group of test activities that are organized and managed together, based on **Level**, **objective**, **Responsibility**, **Time to perform**, **What to test** and others.

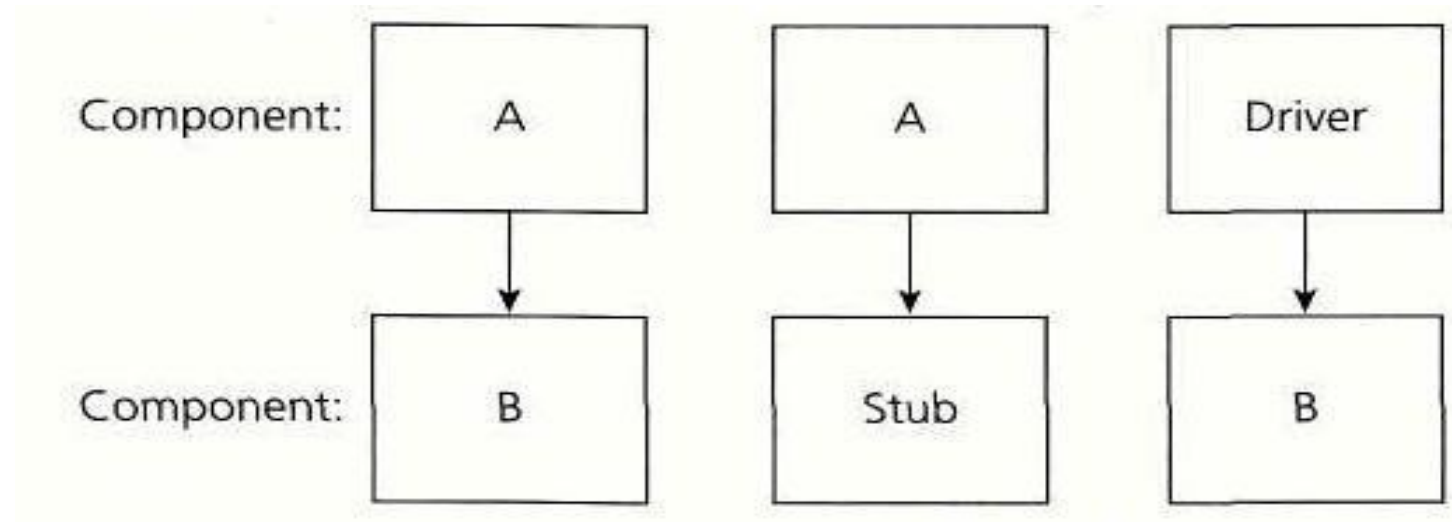






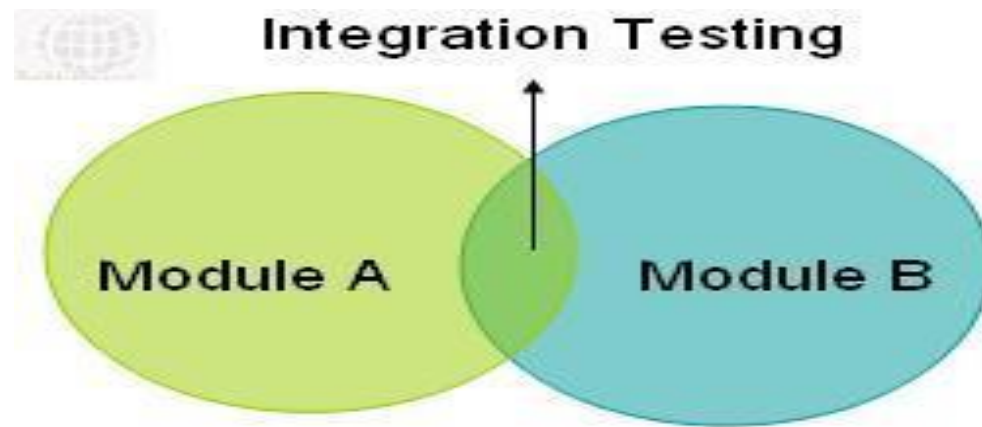
# 1) Unit / Component Testing

- The testing of individual software components that are separate testable (Functions, classes, interfaces, modules)
- Typical test objects for component testing include: Components, units or modules , Code ,data structures, Classes, Database modules.
- **Done by the developer**



## 2) Integration Testing

- The testing performed to expose defects in the **interfaces between components, Interactions to different parts** of a systems and in the Interfaces between systems.
- Test objects for integration testing include: Subsystems, Databases, Infrastructure, Interfaces , and APIs.



# Integration Testing Type

- **Component Integration**

For example, if integrating module A with module B, tests should focus on the communication between the modules, not the functionality of the individual modules, as that should have been covered during component testing.

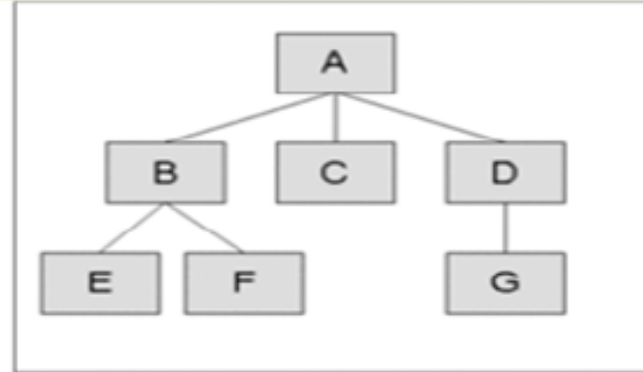
- **System Integration**

If integrating system X with system Y, tests should focus on the communication between the systems, not the functionality of the individual systems, as that should have been covered during system testing.

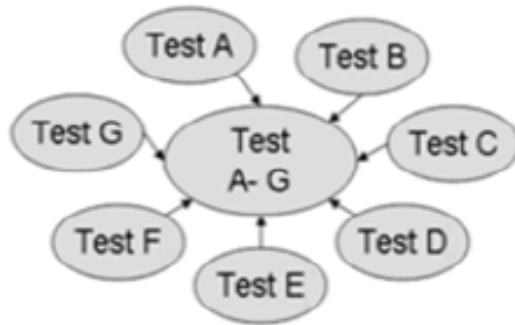
A  
A1  
A2  
A3

B  
B1  
B2  
B3

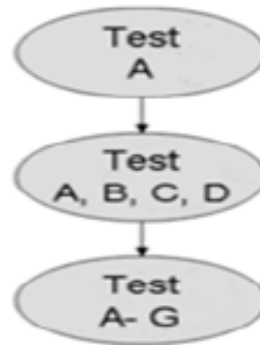
# Integration Approaches -



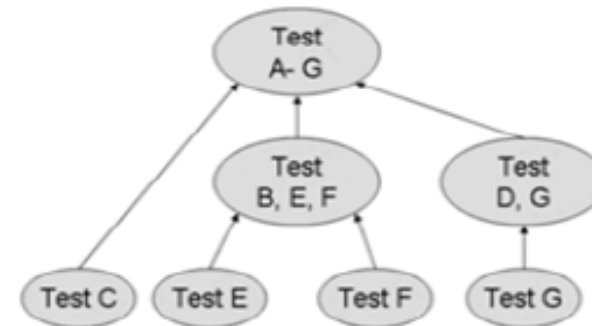
**System Components**



**Big-Bang**



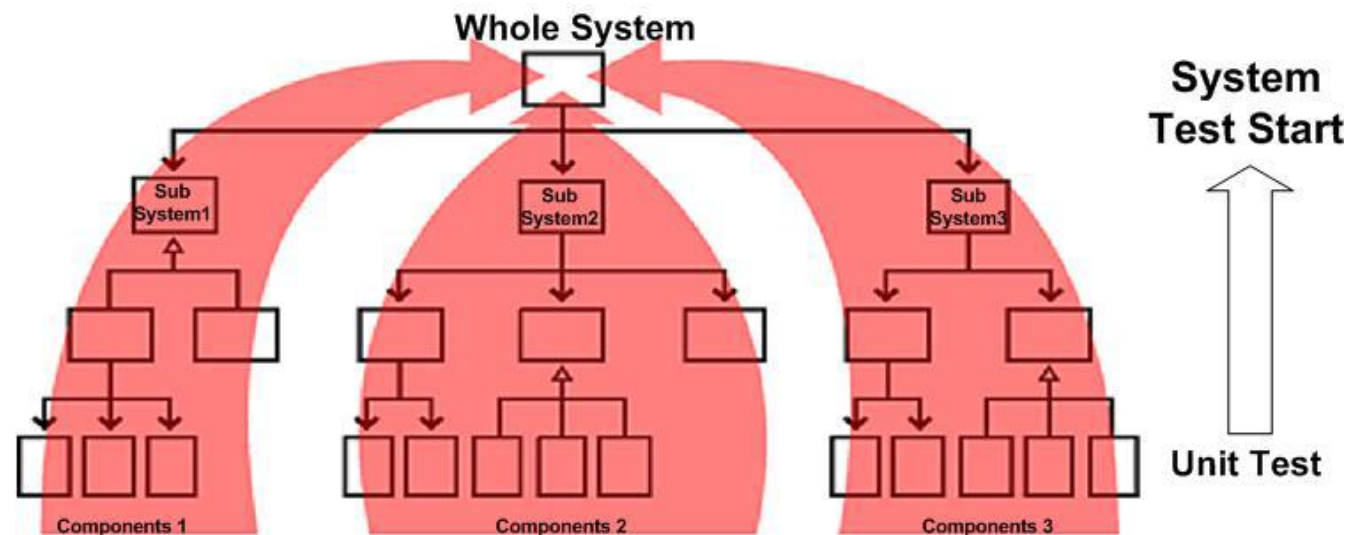
**Incremental  
Top-Down**



**Incremental  
Bottom-Up**

### 3) System Testing

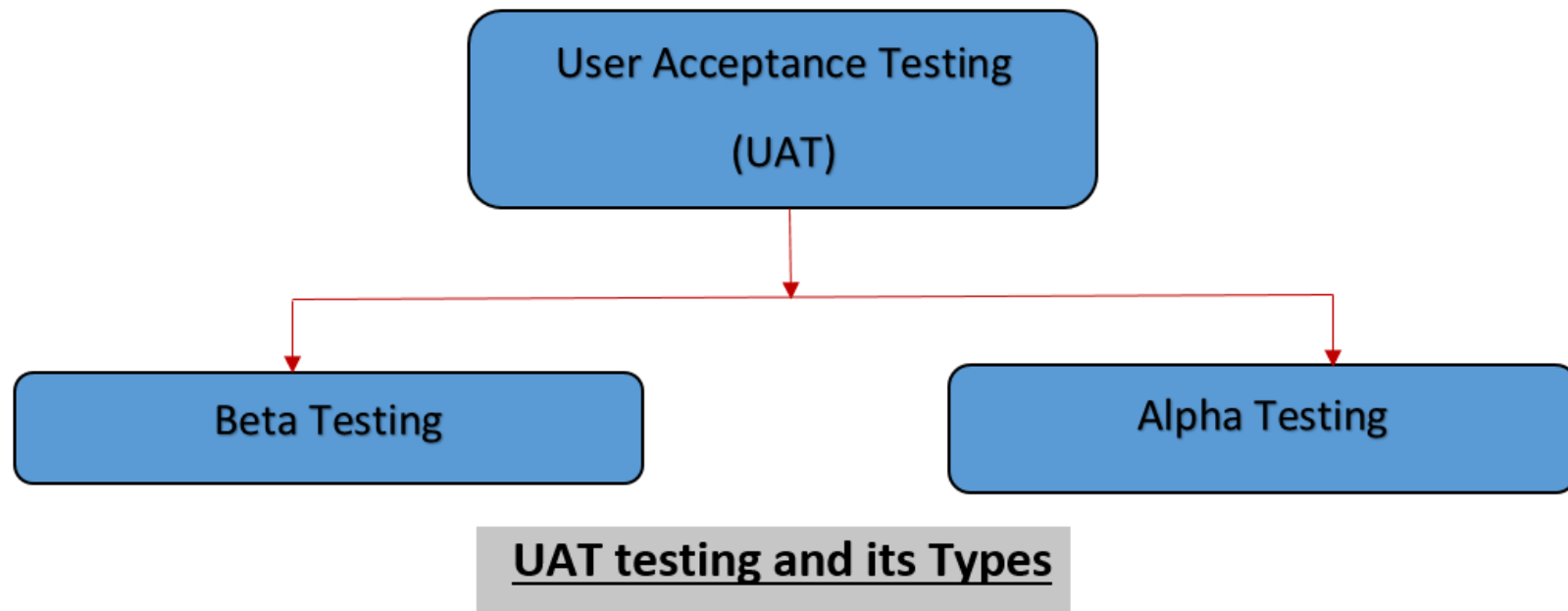
- Testing the behavior of a **whole system** as defined in the **scope of development** project, to verify it **meets the requirements**.
- Typical test objects for system testing include: **Applications**, Hardware/software systems, Operating systems, **System under test**.
- The test environment should correspond to the final target or production environment.
- **Independent testers** typically carry out system testing.



## 4) User Acceptance Testing

### - Validation Testing UAT

- Formal testing with respect to user needs, requirements, and business processes conducted to **determine if a system satisfies the acceptance criteria** and to enable the user, customers or other authorized entity to determine whether or not to accept the system.



# Alpha and Beta testing

- Alpha testing is performed at the developing organization's site by the customers.
- Beta testing is performed by potential or existing customers, and/or operators at their own locations.



**QUIZ  
TIME!**

Which of the following comparisons of component testing and system testing is true?

A. Component testing verifies the functioning of software modules, program objects, and classes that are separately testable, whereas system testing verifies interfaces between components and interactions with different parts of the system

**B. Test cases for component testing are usually derived from component specifications, design specifications, or data models, whereas test cases for system testing are usually derived from requirement specifications, functional specifications or use cases**

C. Component testing focuses on functional characteristics, whereas system testing focuses on functional and non-functional characteristics

D. Component testing is the responsibility of the technical testers, whereas system testing typically is the responsibility of the users of the system

What type of testing is normally conducted to verify that a product meets a particular regulatory requirement?

A. Unit Testing

B. Integration Testing

C. System Testing

**D. Acceptance Testing**

Use cases are a test basis for which level of testing?

A. Unit

**B. System**

C. Load and performance

D. Usability

# Testing Types

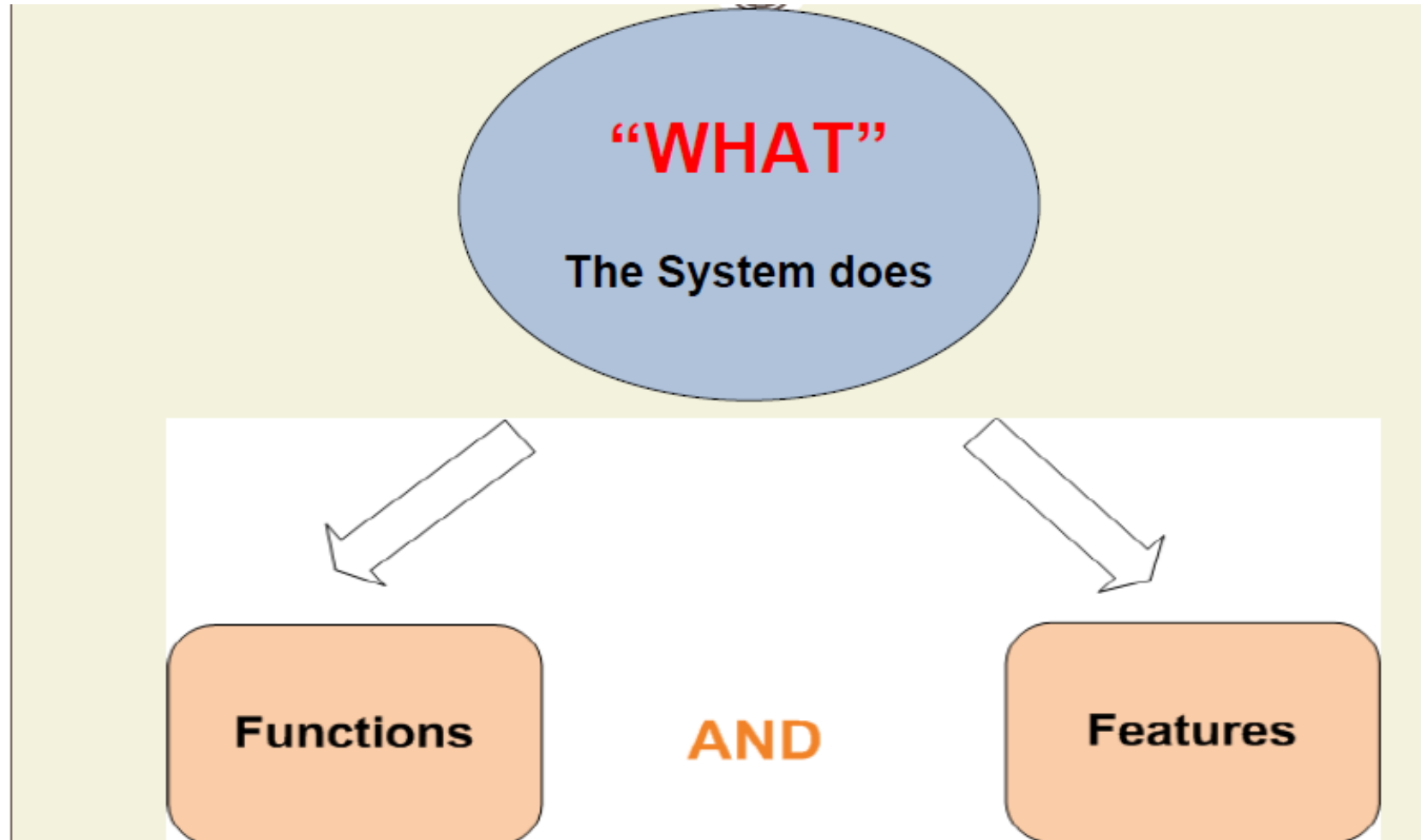
1-Functional Testing

2-Non-functional Testing

3-Structural Testing(White-box Testing)

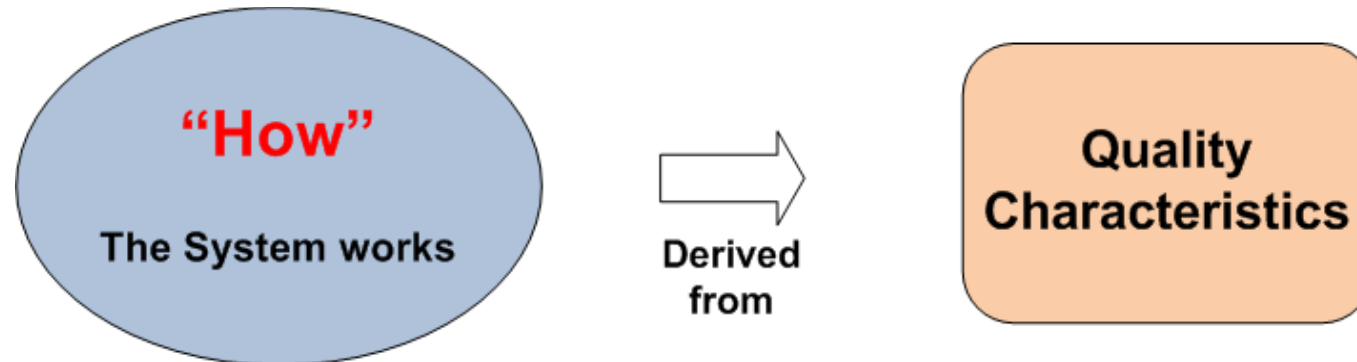
4- Change-related Testing

# 1) Functional Testing



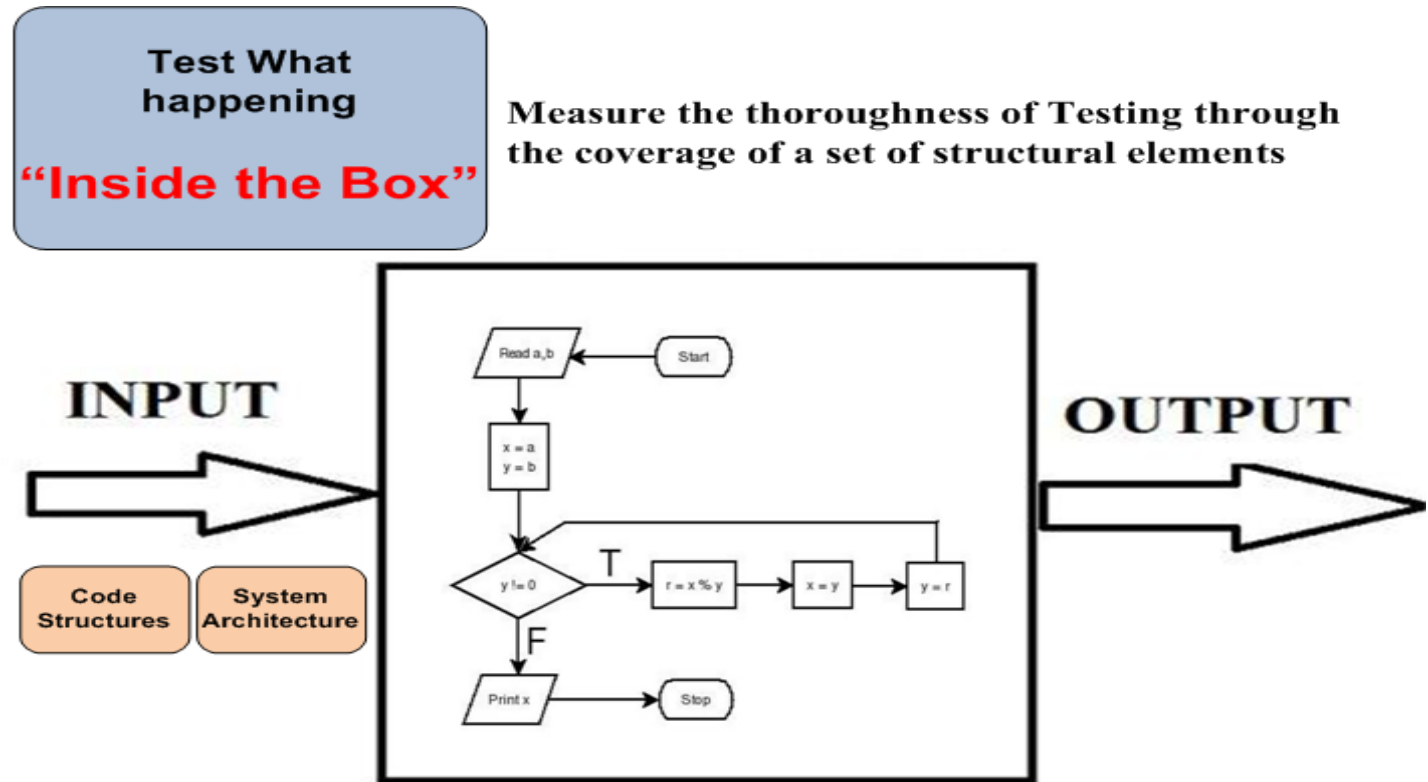
## 2) Non-Functional Testing

- Evaluates **characteristics** of systems and software such as usability, performance efficiency or security.



### 3) Structural Testing (White-box Testing)

- Tests on the system's internal structure.(For loop , If statements)
- Internal structure may include **code**, **architecture**, **work flows**, and/or **data flows** within the system



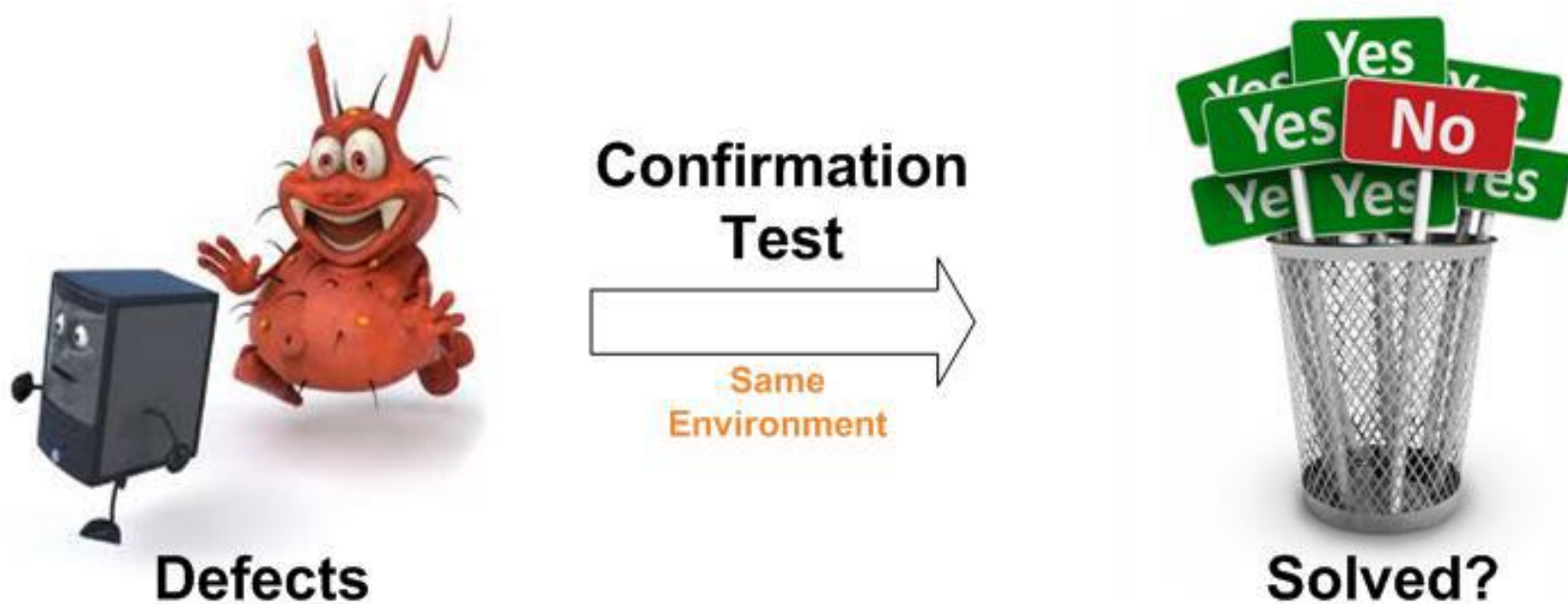


# 4- Change-related Testing

- Confirmation testing(**re-testing**).
- Regression testing

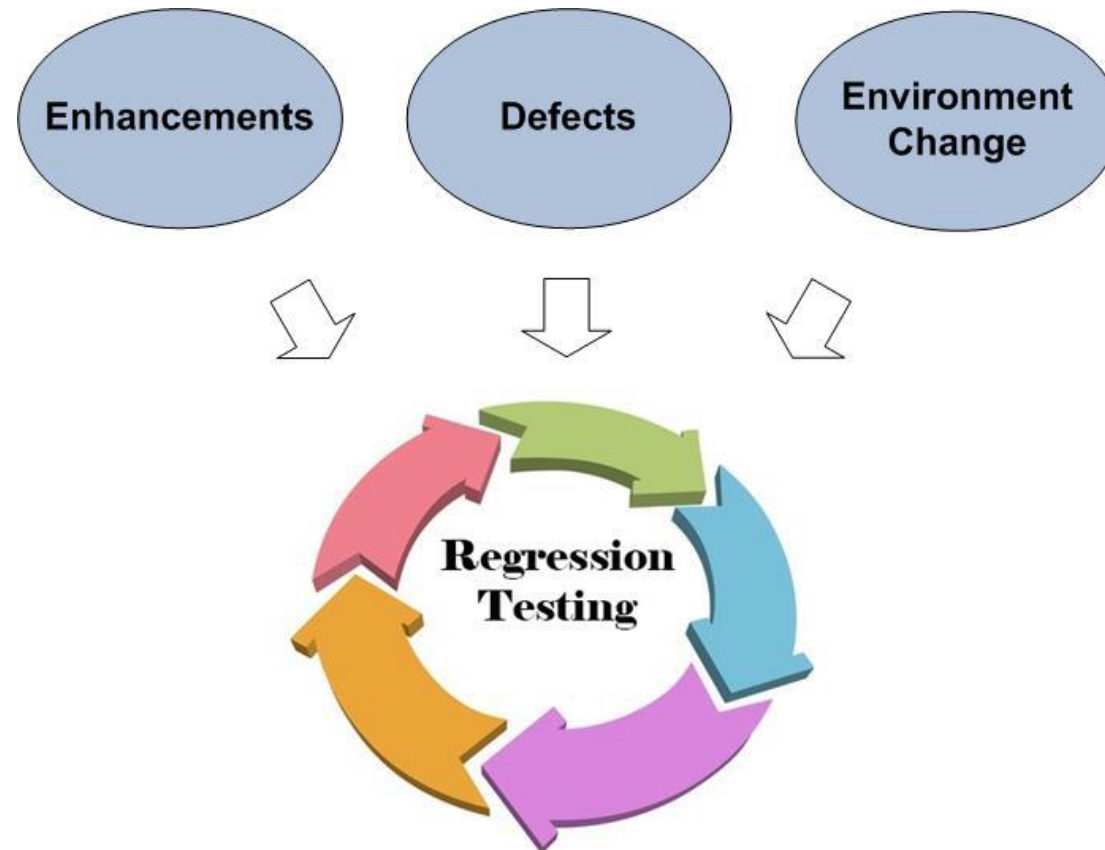
## 4.1 Confirmation testing (Re-Testing)

- After a defect is fixed, the software tested with all test cases that failed due to the defect.



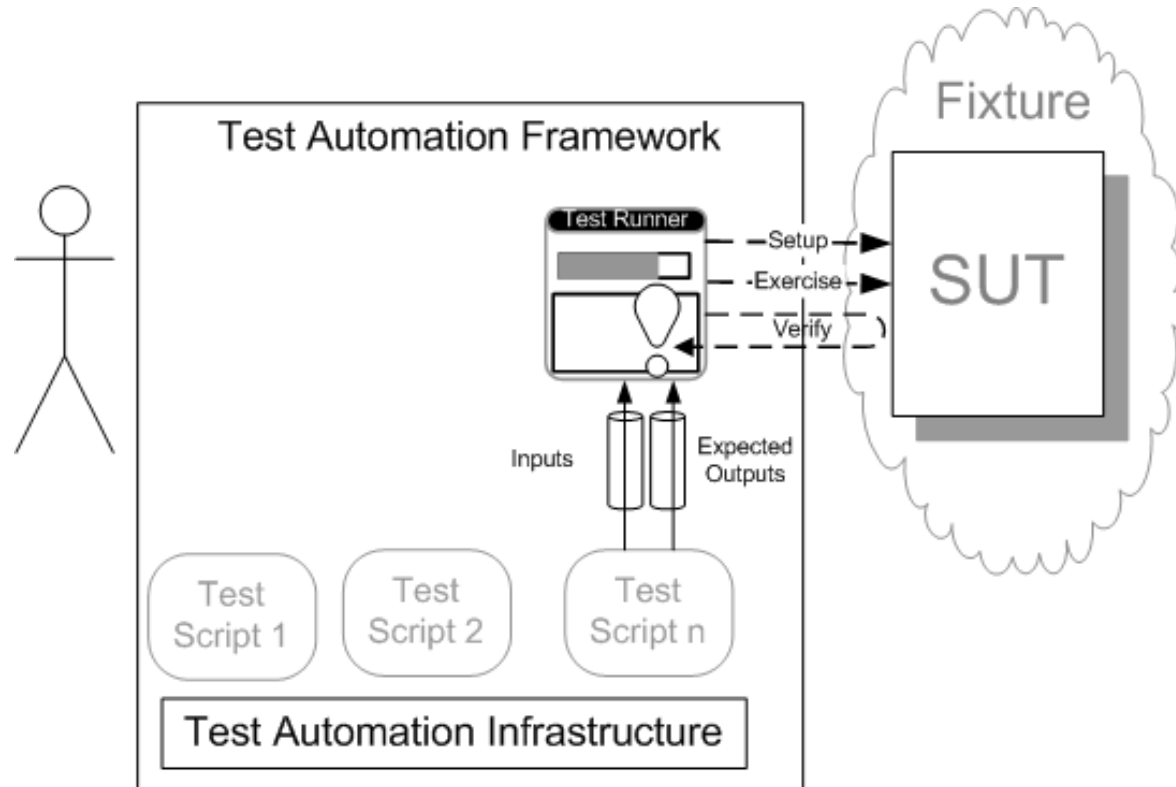
## 4.2 Regression Testing

- It is possible that a change made in one part of the code may affect the other parts of the code.



# Regression Testing Automation

- Regression test suites are run many times so regression testing is a strong candidate for automation.



# Test Types & Test Levels

- It is possible to perform any of the test types mentioned above at any test level.
- **Functional testing** at **component level** -> testing that the code performs its function
- **Non-functional testing** at **component level** -> testing how many cycles(if) does the code use to work
- Functional testing at acceptance level -> testing that users can login successfully to the website
- Non-functional testing at acceptance level -> collect users feedback about the usability of the application

# Maintenance Testing

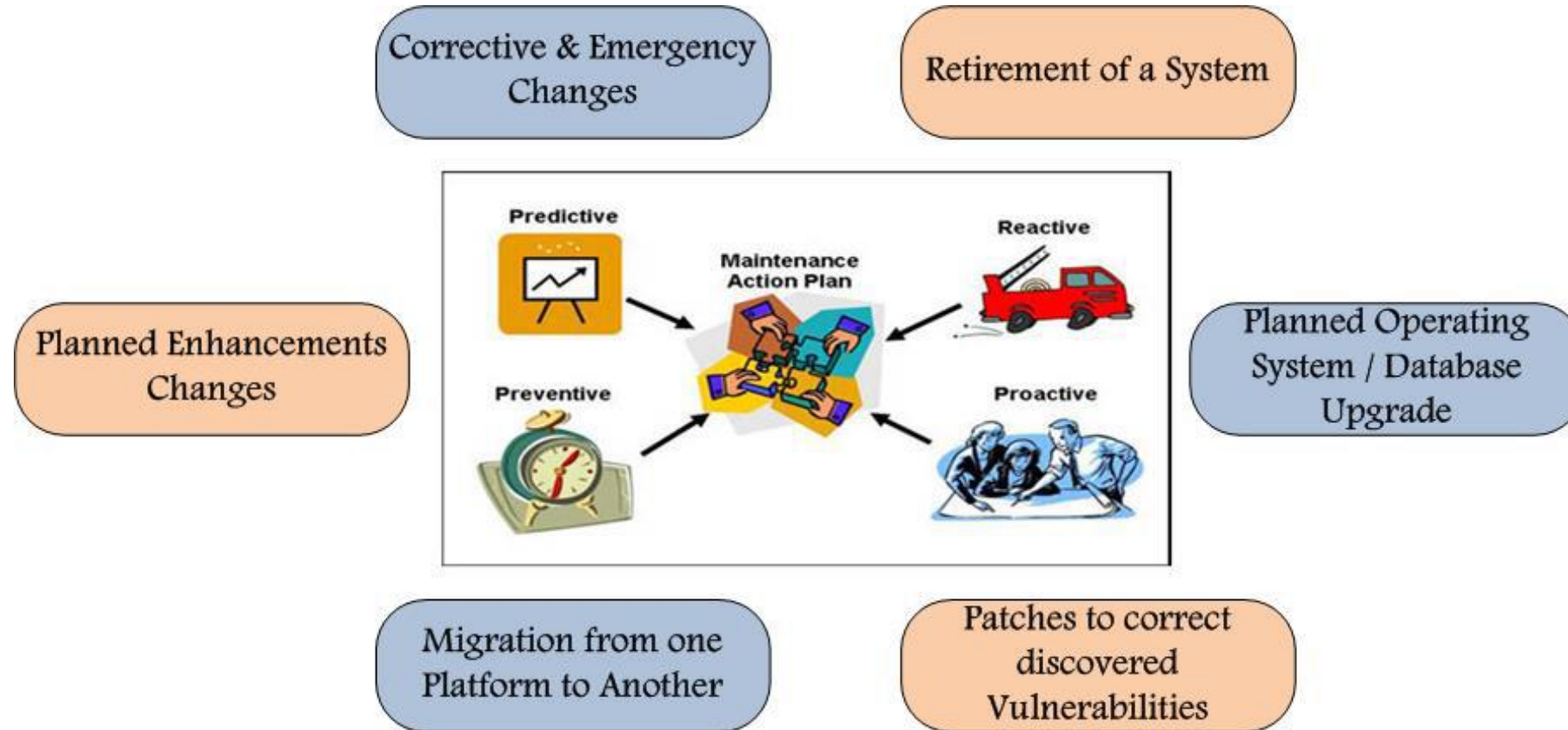
- Any change after delivered software and go live.
- Testing the Changes to the Operational System or the impact of the Changed Environment.  
CR ->Change request
- Depending on Size & Risk of Changes ... Several Levels of Testing will be executed.
- Maintenance can involve **planned releases** and **unplanned releases** (hot fixes)



# Type of Maintenance – Not related ISTQB CTFL

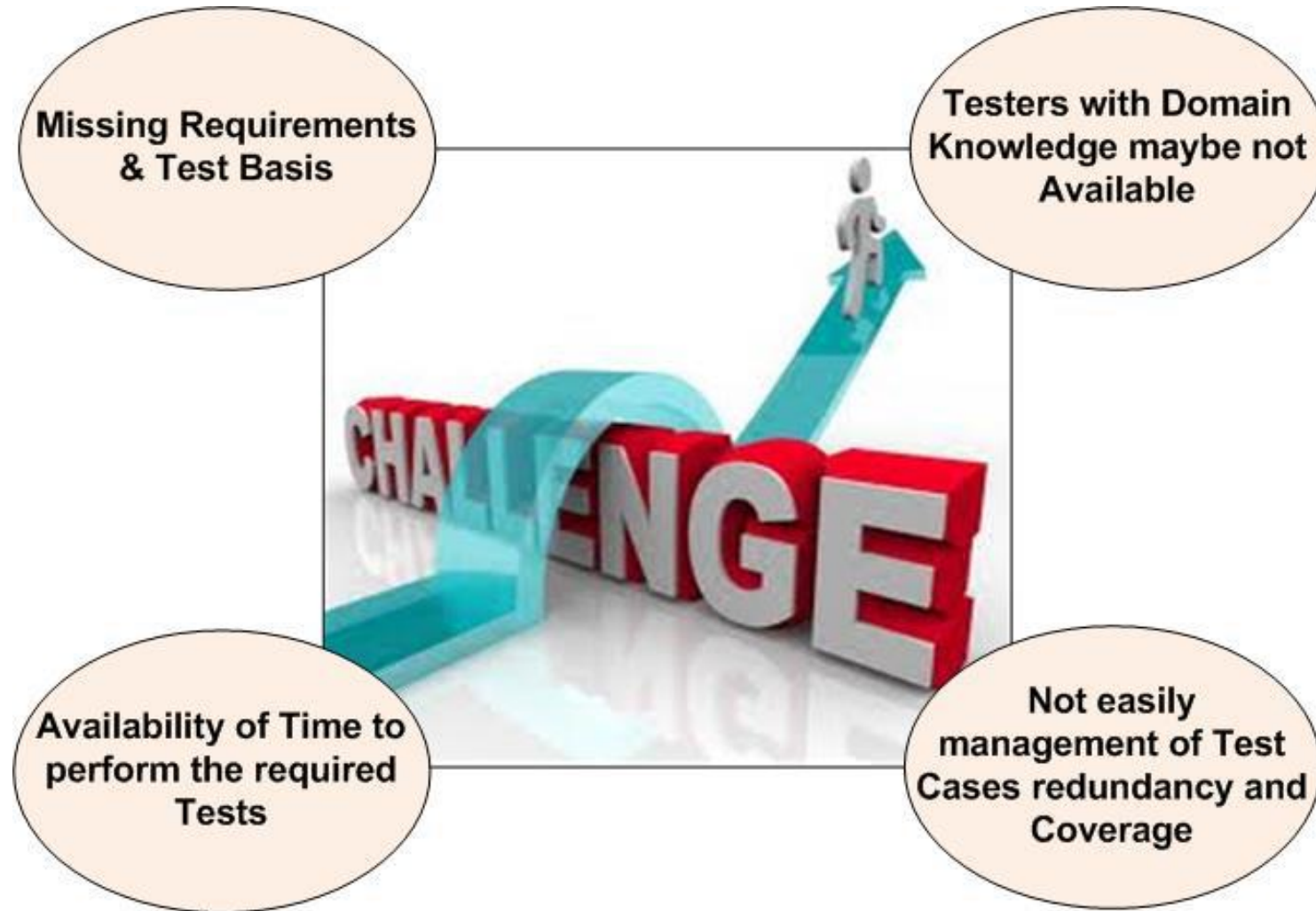
- **Corrective Maintenance** : modifications done in order to correct or fix defect.
- **Adaptive Maintenance** : This includes modifications applied to keep the software product technology up-to date e.g. Update Library , SDK, change programming language from C to C+ [**Without change on software functionality**].
- **Perfective Maintenance**: This includes modifications to add new features, new user requirements.
- **Preventive Maintenance**: This includes modifications to prevent future problems of the software.

# Example of Changes





# Challenges

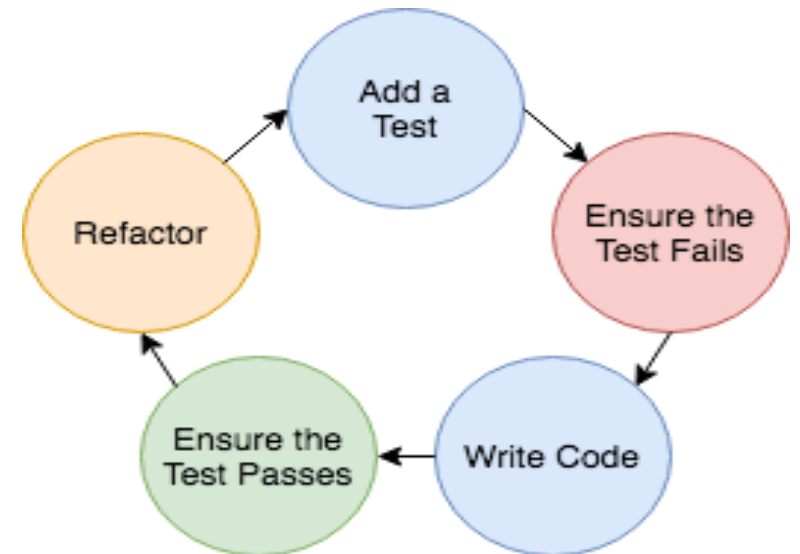


Project > Epic > User Story > Testing Task  
    > Valid & invalid Test Case > bug report

## V4: Testing as a Driven for Software Development

# 1. Test-Driven Development (TDD)

- Test-driven development (TDD) is used to develop code guided by automated test cases
- TDD is a “test first” approach to develop reusable automated tests
- TDD helps to document the code for future maintenance efforts
- Done by **developer** .



# Unit Testing

- Step 1: write logic

```
double calculateLoan (int amount, int  
months, float interestRate)
```

```
{
```

```
    // ...assume functionality
```

```
    // ...
```

```
}
```

- Step 2: write test(s)

```
calculateLoan(2000,60,5.0)
```

```
// is result 382.02?
```

```
calculateLoan(6000,12,10.0)
```

```
// is result 572.50?
```

# TDD

- Step 1: write test(s) first

```
calculateLoan(2000,60,5.0)
// is result 382.02?
```

```
calculateLoan(6000,12,10.0)
// is result 572.50?
```

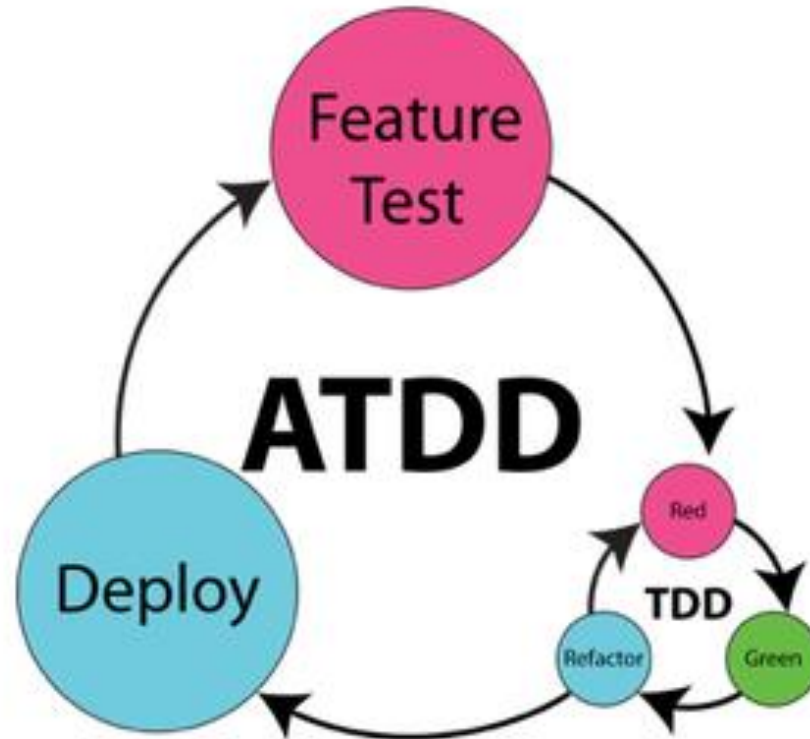
- Step 2: write logic

```
double calculateLoan (int amount, int
months, float interestRate)
{
    // ...assume functionality
    // ...
}
```

## 2.Acceptance Test-Driven Development (ATDD)

- Acceptance test-driven development defines acceptance criteria and tests during the creation of user stories.
- Acceptance test-driven development is a collaborative approach that allows every stakeholder to understand how the software component has to behave and what the **developers, testers, and business** representatives need to ensure this behavior.
- ATTD **reusable tests for regression testing.**
- ATTD is test –first approach , **Test Cases created before implementing user story.**
- Test cases are created by **agile team**
- Natural language so can all understand it.

# Acceptance Test-Driven Development (ATDD)





## ATDD-Example

- If customer Rating is good and the order total is less than or equal to \$10.00,  
Then do not give a discount  
Otherwise give a 1% discount
- If customer rating is Excellent,  
Then give a discount of 1 % for any order
- If the order total is greater than \$50.00  
Then give a discount of 5%

**What should we do with good customer with order \$50.01**

## ATDD-Example

| Discount    |                 |                     |
|-------------|-----------------|---------------------|
| Order Total | Customer Rating | Discount Percentage |
| 10.00       | Good            | 0                   |
| 10.01       | Good            | 1                   |
| 50.01       | Good            | 1                   |
| 0.01        | Excellent       | 1                   |
| 50.00       | Excellent       | 1                   |
| 50.01       | Excellent       | 5                   |

## ATDD-Example

```
class TestCase {  
    testDiscountPercentageForCustomer() {  
        someClass x = new someClass();  
        assertEquals (0, x.computeDiscount (10.0, Good));  
        assertEquals (1, x.computeDiscount (10.01, Good));  
        assertEquals (1, x.computeDiscount (50.01, Good));  
        assertEquals (0, x.computeDiscount (.01, Excellent));  
        assertEquals (0, x.computeDiscount (50.0, Excellent));  
        assertEquals (0, x.computeDiscount (50.01, Excellent));  
    }  
}
```

# 3. Behavior-Driven Development (BDD)

- Behavior-driven development allows developer to focus on testing the code based on the **expected behavior** of the software.
- From these requirements, the behavior-driven development framework generates code that can be used by developers to create test cases.

**Given** some context(Starting State),

**When** an event (What the user dose) occurs,

**Then** ensure some outcomes(Expected Result).

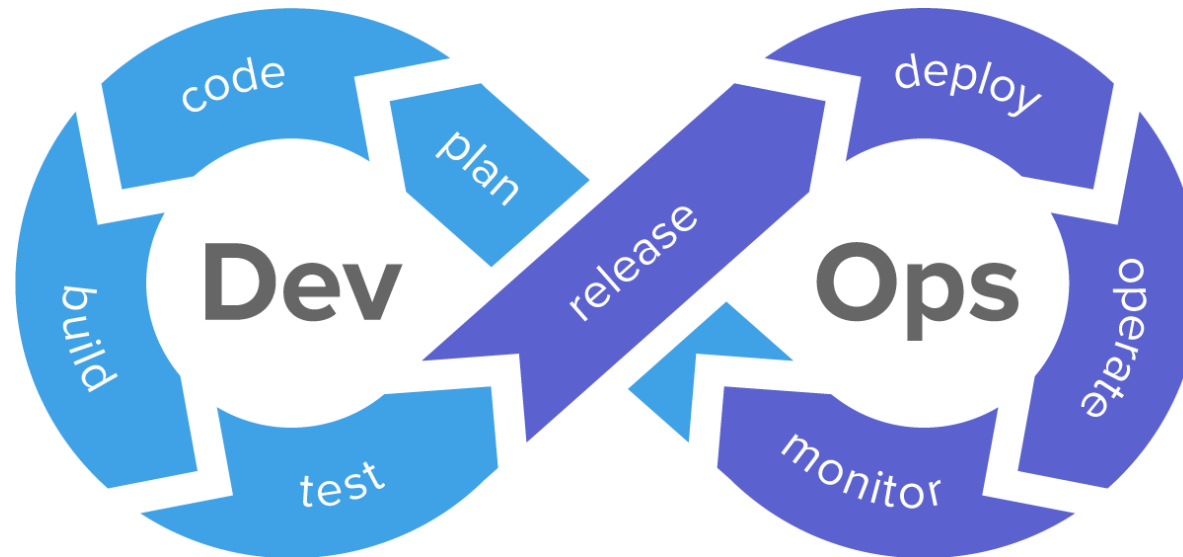
## BDD ATM Machine Example

- Given the account balance is \$100  
And the card is valid  
And the machine contains enough money
- When the account holder requests \$20
- Then the cashpoint should dispense \$20  
And the account balance should be \$8



# DevOps >> to build CI vs. CD

- Development and operations teams use to build, test, deploy, and monitor applications with speed, quality, and control.



# benefits of DevOps

- Fast feedback on the code quality.
- CI promotes a shift-left approach in testing by encouraging developers to **submit high quality code** accompanied by component tests and static analysis. [ Ch3]
- Promotes automated processes like CI/CD that facilitate establishing **stable test environments**.
- Increases the view on non-functional quality characteristics (e.g., performance, reliability)
- Automation through a delivery pipeline reduces the need for repetitive manual testing.
- **The risk in regression is minimized due to the scale and range of automated regression tests**

# DevOps risks and challenges

- The DevOps delivery pipeline must be defined and established
- CI / CD tools must be introduced and maintained
- Test automation requires additional resources and may be difficult to establish and maintain



# Shift-Left Approach

- The principle of **early testing** is sometimes referred to as shift-left because it is an approach where testing is performed **earlier in the SDLC**. Shift-left normally suggests that testing should be done earlier

## shift-left include:

Reviewing the specification from the perspective of testing.

Writing test cases before the code is written.

Using CI and even better CD as it comes with fast feedback.

Completing **static analysis** of source code.

Performing non-functional testing starting at the component test level, where possible.

# Retrospectives and Process Improvement

- Retrospectives (also known as “**post-project meetings**” and project retrospectives) are often held at the end of a **project** or an **iteration**.
- What was **successful**, and should be retained?
- What was not successful and could be **improved**?
- How to improvements and retain the successes in the future? (**Action items**).

# Retrospectives benefits for testing include

- Increased test effectiveness / efficiency (e.g., by implementing suggestions for process improvement)
- Increased quality of testware and test basis.
- Team learning
- Better cooperation between development and testing.

**QUIZ  
TIME!**

Which one of the following is TRUE?

- A. The purpose of regression testing is to check if the correction has been successfully implemented, while the purpose of confirmation testing is to confirm that the correction has no side effects
- B. The purpose of regression testing is to detect unintended side effects, while the purpose of confirmation testing is to check if the system is still working in a new environment
- C. The purpose of regression testing is to detect unintended side effects, while the purpose of confirmation testing is to check if the original defect has been fixed
- D. The purpose of regression testing is to check if the new functionality is working, while the purpose of confirmation testing is to check if the originally defect has been fixed.

Which of the following is most correct regarding the test level at which functional tests may be executed?

- A. Unit and integration
- B. Integration and system
- C. System and acceptance
- D. All levels**

Usability testing is an example of which type of testing?

A. Functional

**B. Non-functional**

C. Structural

D. Change-related

You have been receiving daily builds from the developers. Even though they are documenting the fixes they are including in each build, you are finding that the fixes either aren't in the build or are not working. What type of testing is best suited for finding these issues?

- A. Unit Testing
- B. System Testing
- C. Confirmation Testing**
- D. Regression Testing



During which level of testing should non-functional tests be executed?

- A. Unit and integration only
- B. System testing only
- C. Integration, system and acceptance only
- D. Unit, integration, system and acceptance only**

If impact analysis indicates that the overall system could be significantly affected by system maintenance activities, why should regression testing be executed after the changes?

- A. To ensure the system still functions as expected with no introduced issues**
- B. To ensure no unauthorized changes have been applied to the system
- C. To assess the scope of maintenance performed on the system
- D. To identify any maintainability issues with the code