SOFTWARE TESTING

INTRODUCTION

Dr. ARIVUSELVAN.K

Associate Professor – (SCORE)

VIT University

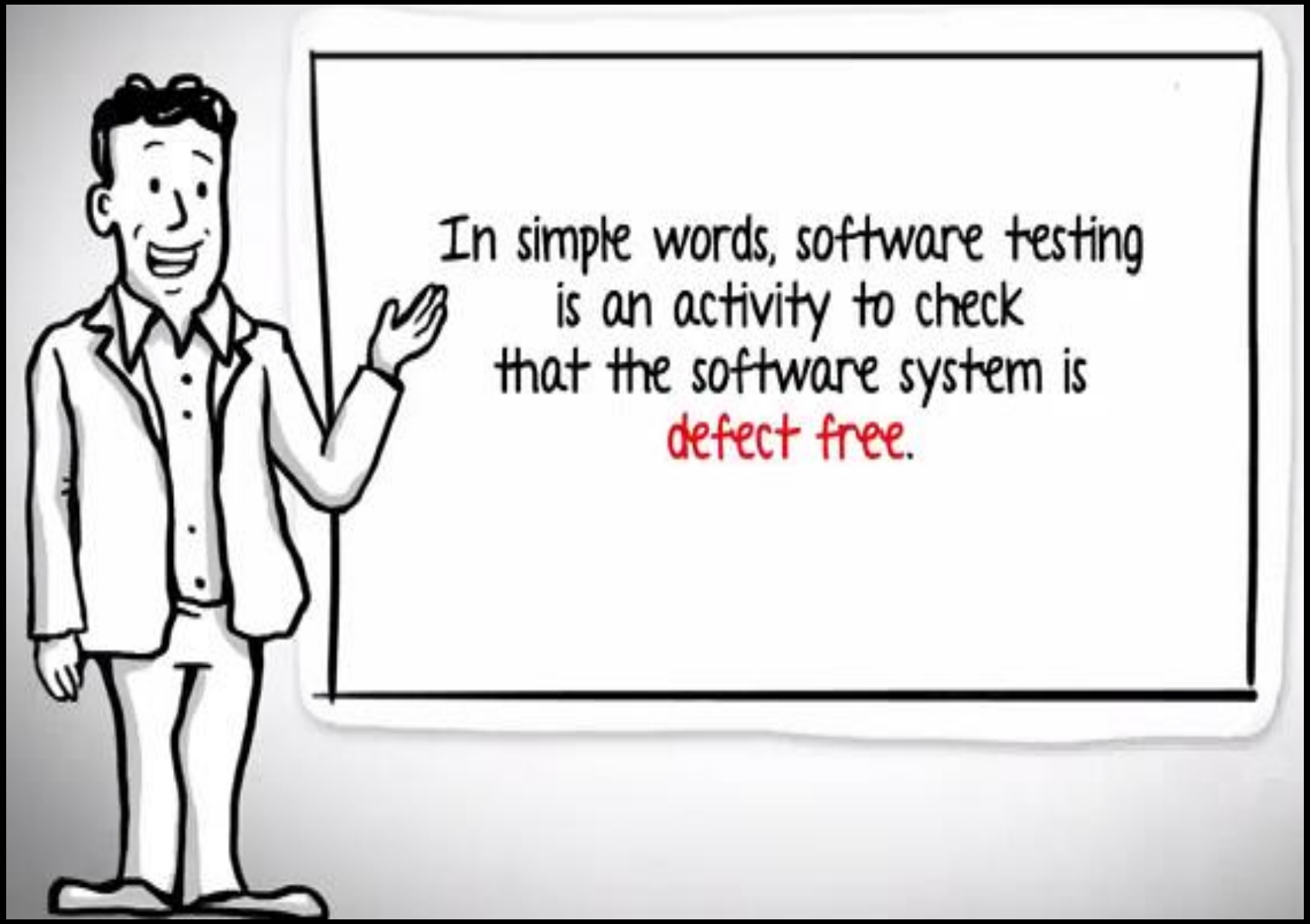
Module One -Objectives

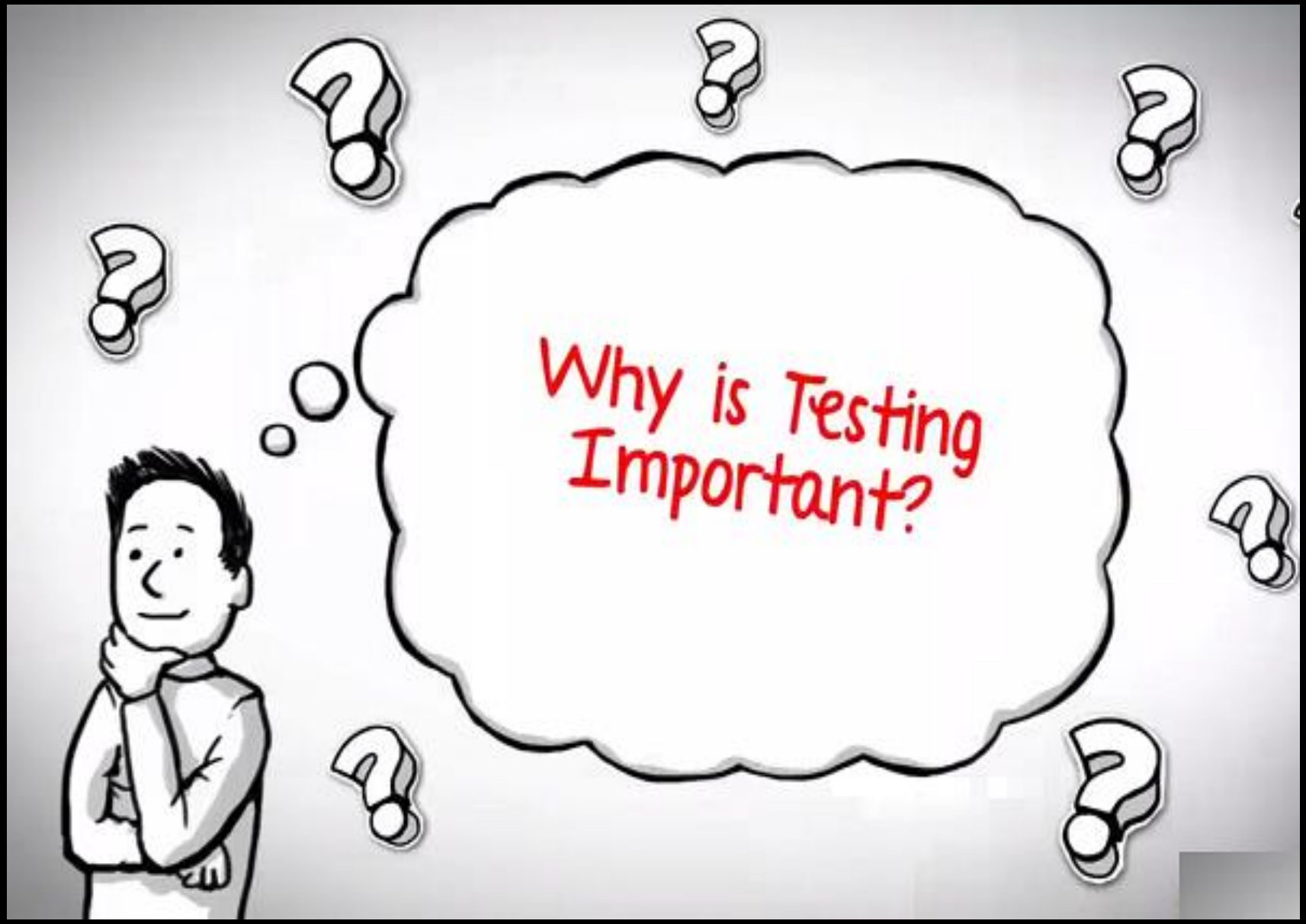
- How the software testing has been evolved?
- What are software testing myths and what is the corresponding truth?
- What are the goals of Software testing?
- Various Principles of Software testing.

SOFTWARE TESTING

What is Software Testing?

Software testing is a process used to identify the correctness,completeness and quality of developed computer software.It includes a set of activities conducted with the intent of finding errors in software so that it could be corrected before the product is released to the end users.



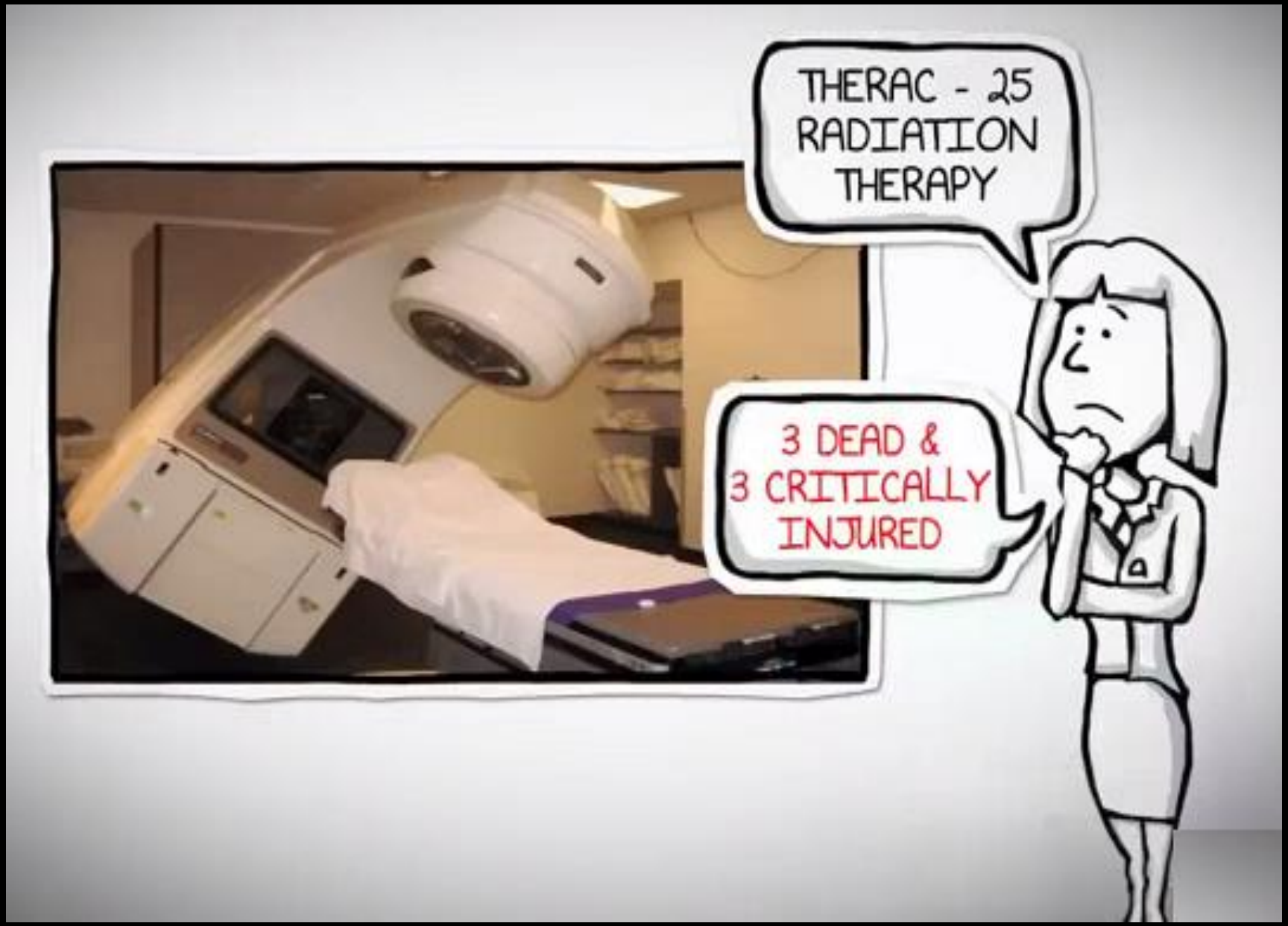


Software Bugs can be
Expensive or even
Dangerous

China Airbus A300-26th April 1994

Airplane Crash

264 People dead

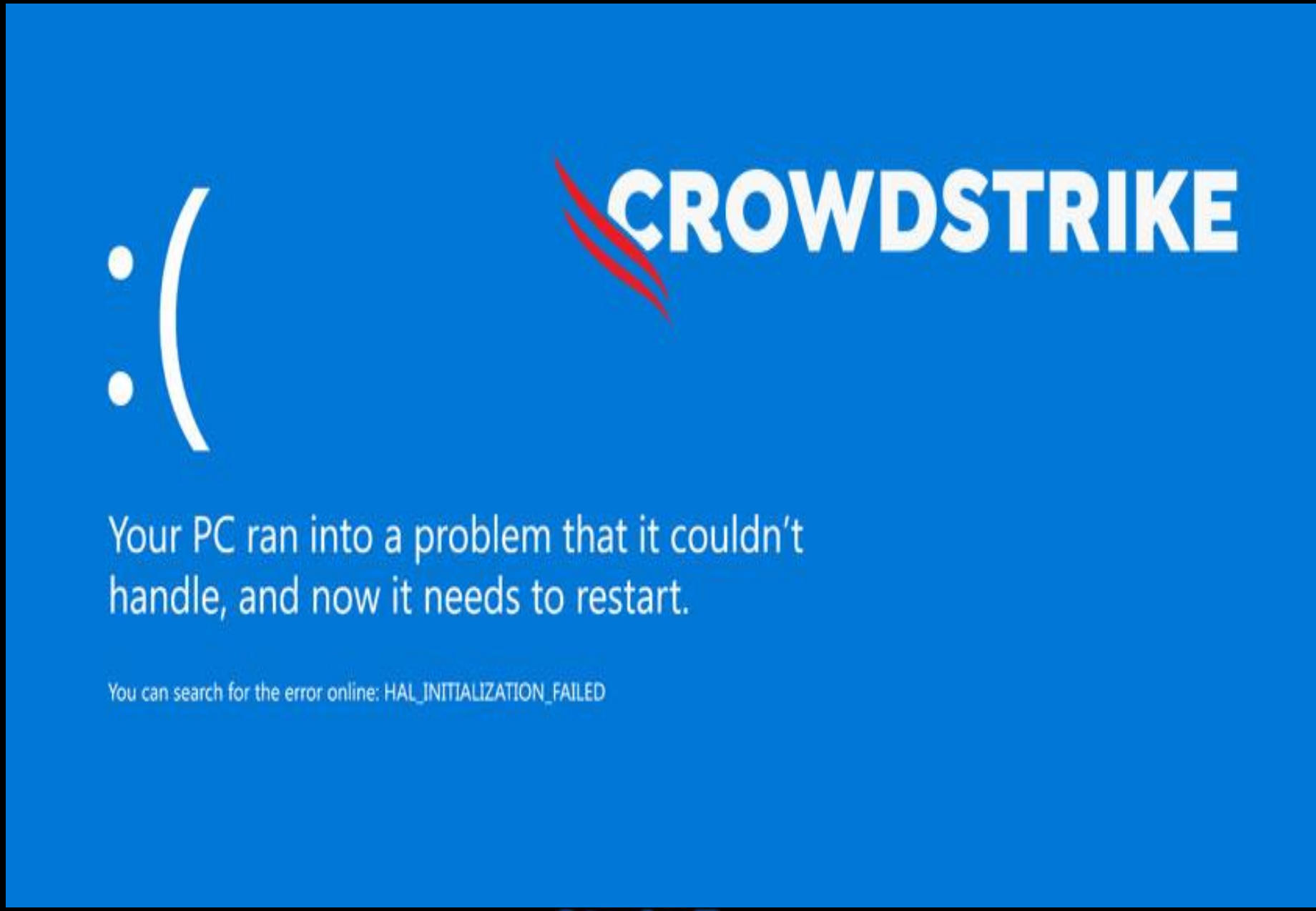


Failed Satellite Launch

$ 1.2 billion lost

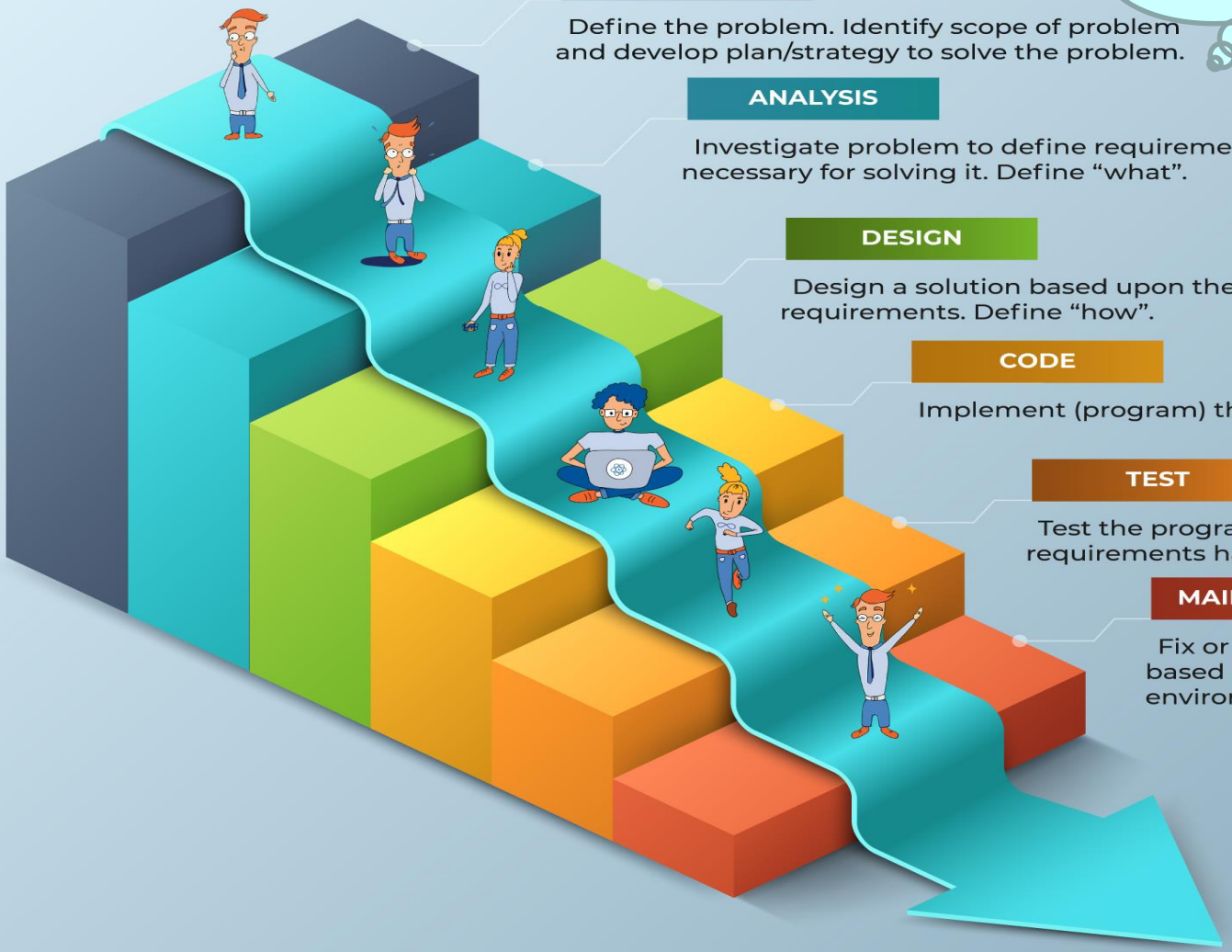
U.S. Bank Accounts

823 Customers paid $920 million



How a SOFTWARE is developed?

Using SDLC (Waterfall / AGILE)



REQUIREMENTS

Define the problem. Identify scope of problem and develop plan/strategy to solve the problem.

ANALYSIS

Investigate problem to define requirements necessary for solving it. Define “what”.

DESIGN

Design a solution based upon the requirements. Define “how”.

CODE

Implement (program) the design.

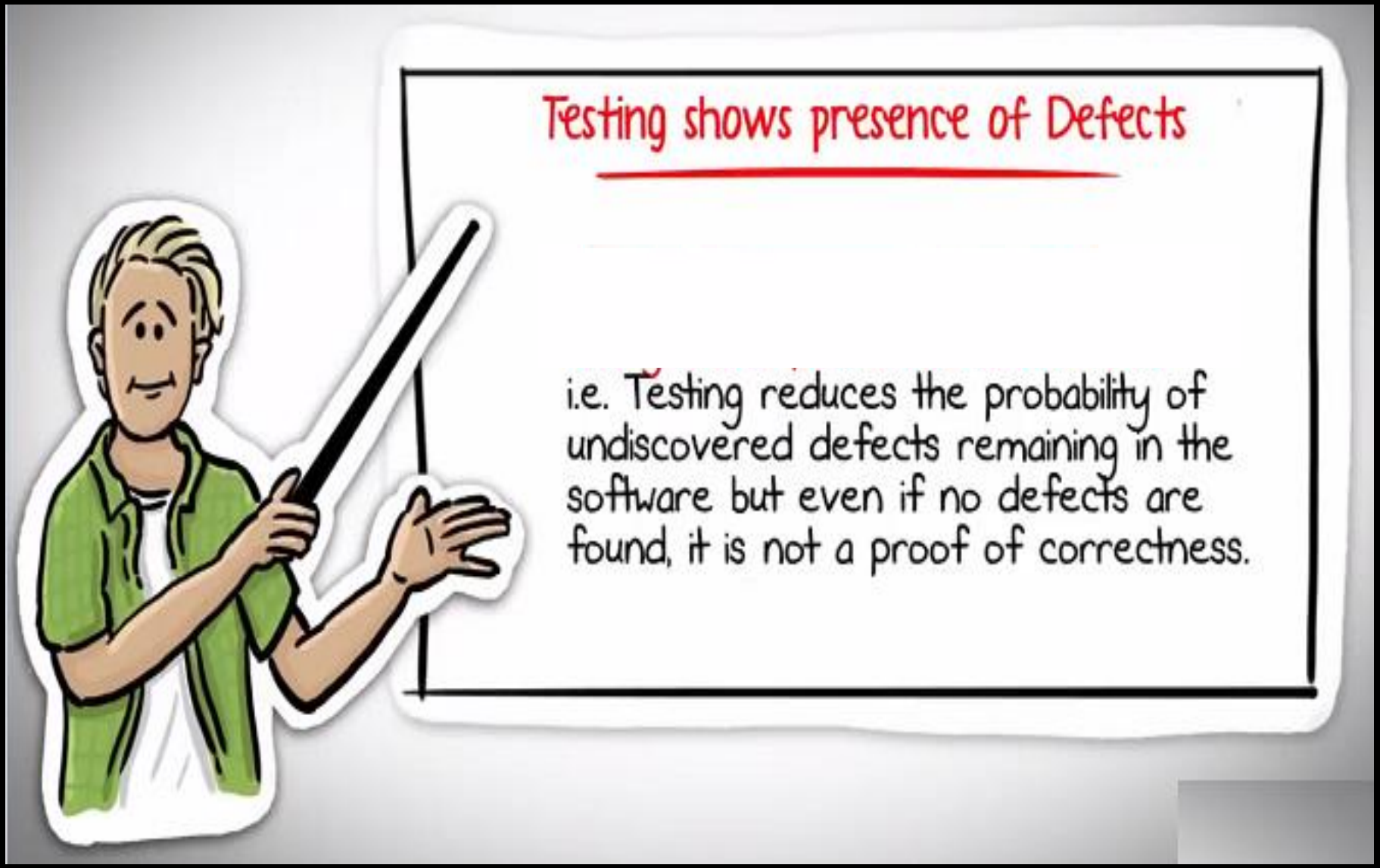
TEST

Test the program to ensure the requirements have been satisfied.

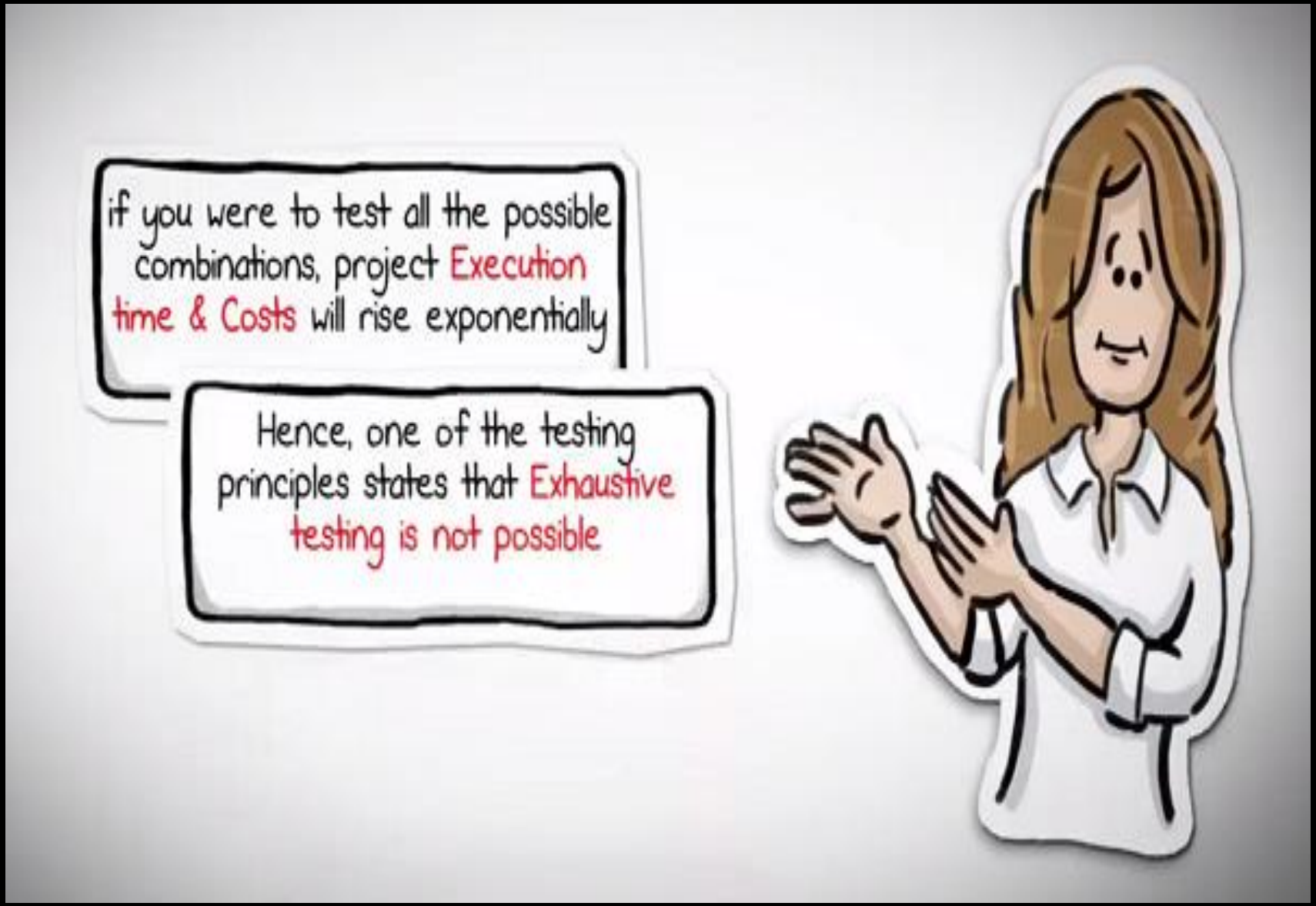
MAINTENANCE

Fix or improve applications based upon use or changes in the environment.

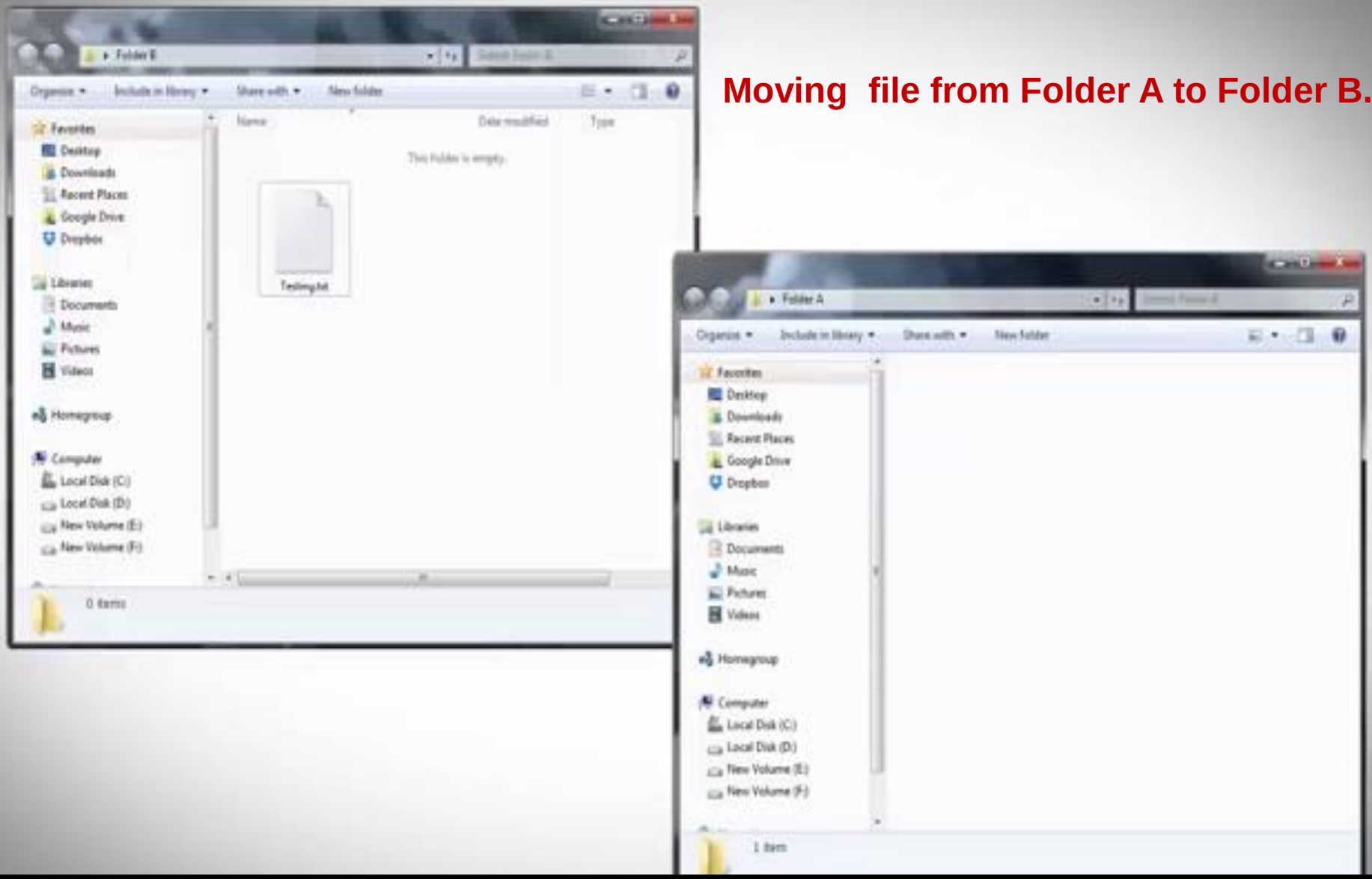
SEVEN PRINCIPLES (SOFTWARE TESTING)

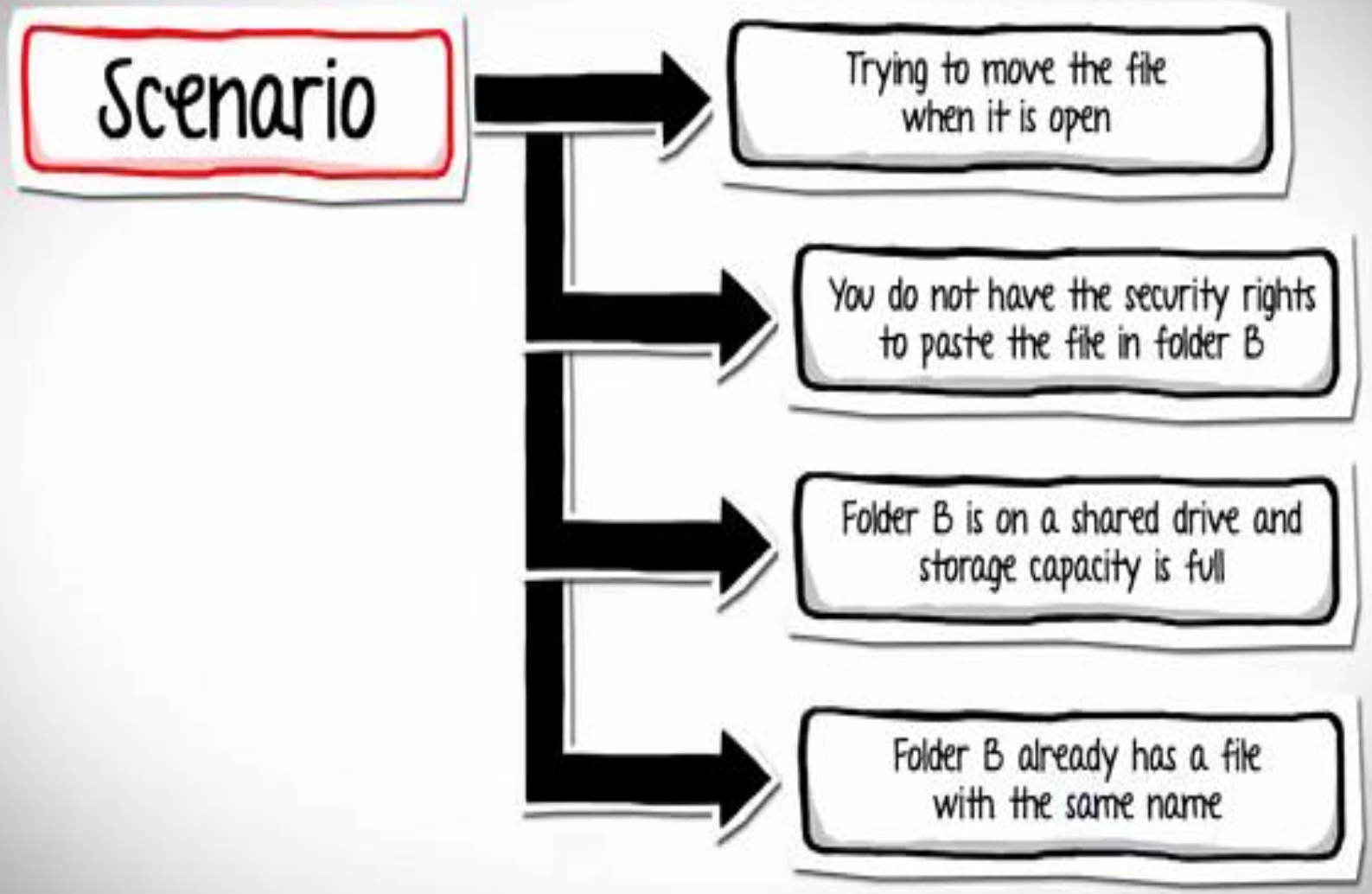


if you were to test all the possible combinations, project Execution time & Costs will rise exponentially



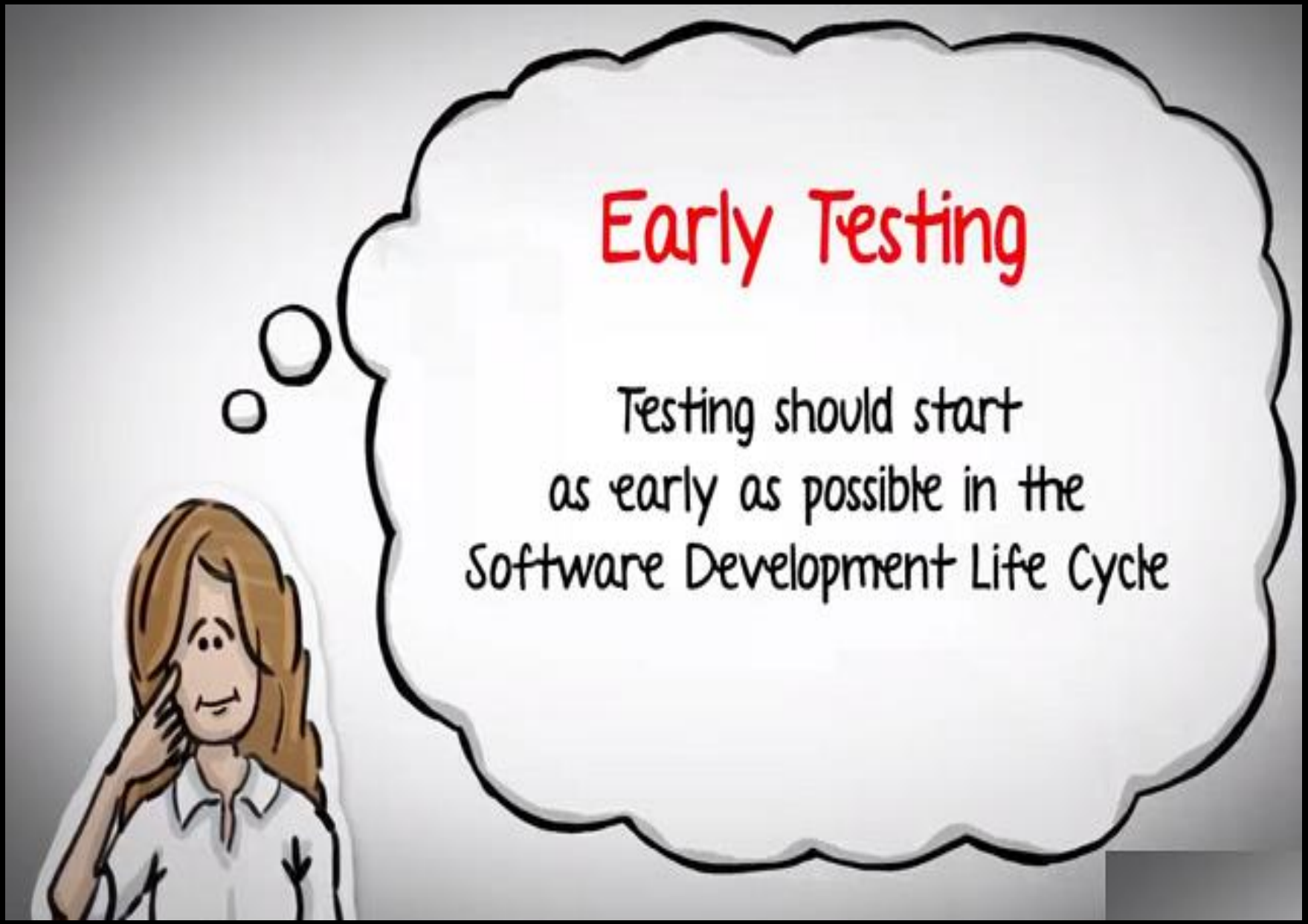
Moving file from Folder A to Folder B.

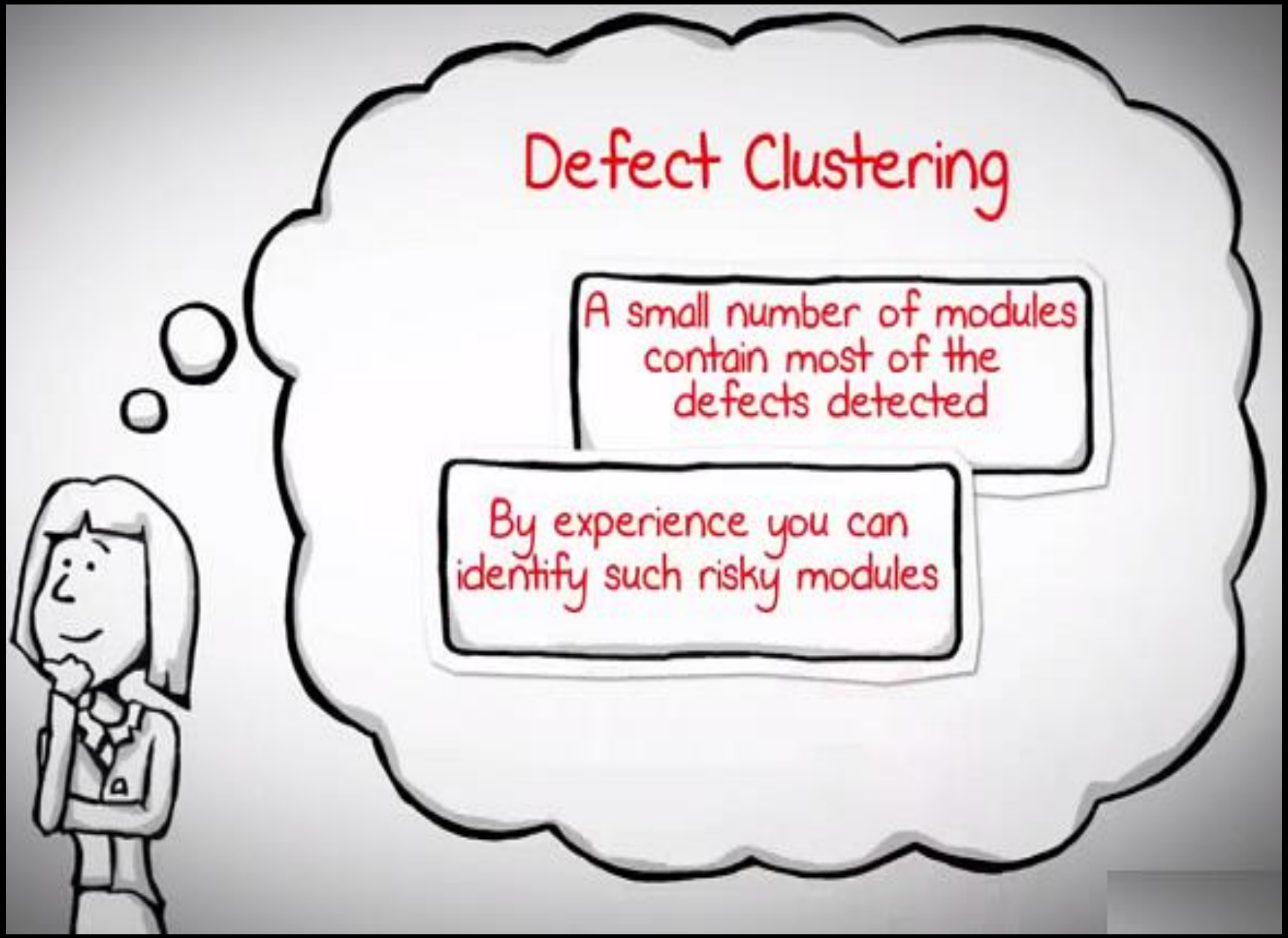




Or suppose you have 15 input fields to test each having 5 possible values, the number of combinations to be tested would be 5^15 = 30 517 578 125!!

Field 1	Field 2	Field 3
Field 4	Field 5	Field 6
Field 7	Field 8	Field 9
Field 10	Field 11	Field 12
Field 13	Field 14	Field 15

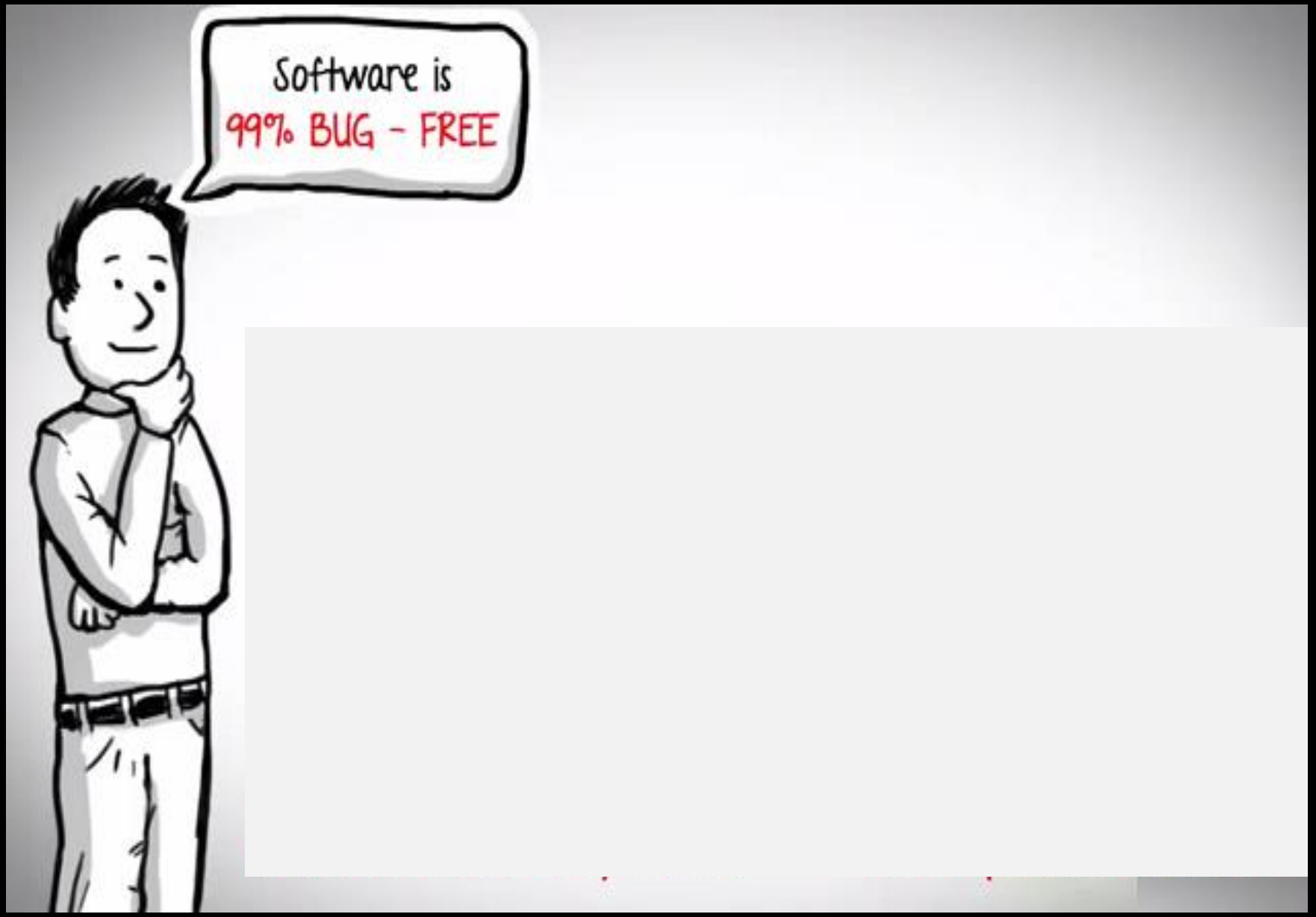




- If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs
- This is the another priciple of testing called "Pesticide paradox"
- To overcome this, the test cases needed to be regularly reviewed & revised, adding new and different test cases to help find more defects.

Testing is Context dependent

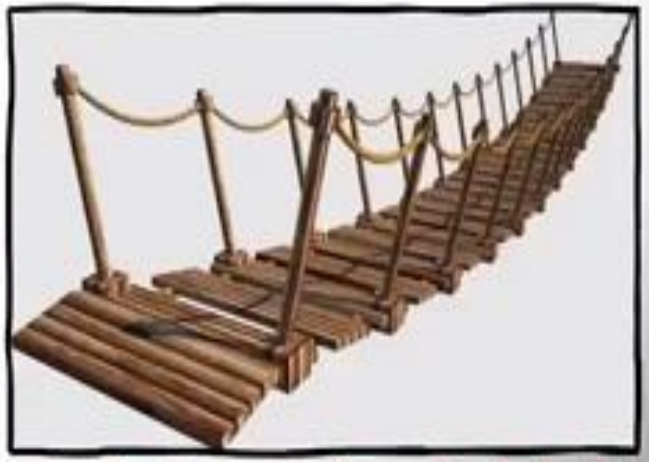
Windows 98
crashes live on CNN



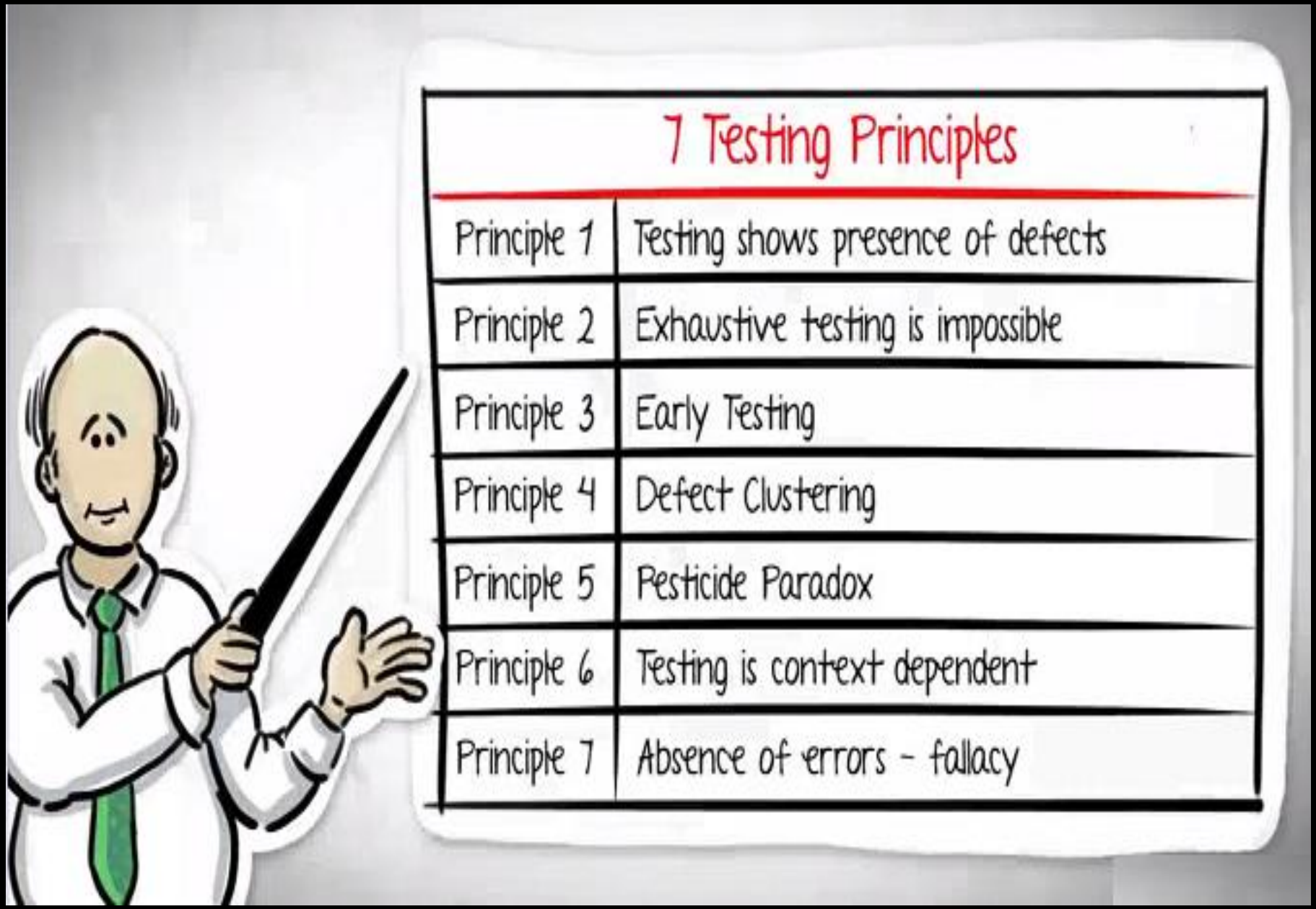
Absence of error is a fallacy

Software does not meet the needs and requirements of the client

What the client really needed



What operations installed



SOFTWARE TESTING— MYTHS AND FACTS

Myth 1: Testing is a single phase in SDLC .

Truth: Testing starts as soon as we get the requirement specifications for the software.

And the testing work continues throughout the SDLC, even post-implementation of the software.

SOFTWARE TESTING— MYTHS AND FACTS

Myth 2: Testing is easy.

Truth: Testers' job is not easy, as they have to plan and develop the test cases manually and it requires a thorough understanding of the project being developed with its overall design.

Testers have to shoulder a lot of responsibility which sometimes make their job even harder than that of a developer.

SOFTWARE TESTING— MYTHS AND FACTS

Myth 3: Software development is worth more than testing.

We have this myth right from the beginning that, testing is considered a secondary job.

Truth: Testing is a complete process like development, so the testing team enjoys equal status and importance as the development team.

SOFTWARE TESTING— MYTHS AND FACTS

Myth 4: Complete testing is possible.

Truth: It is not possible to provide all the possible inputs to test the software, as the input domain of even a small program is too large to test.

This is the reason why the term 'complete testing' has been replaced with 'effective testing'.

SOFTWARE TESTING— MYTHS AND FACTS

Myth 5: Testing starts after program development.

Truth: The tester performs testing at the end of every phase of SDLC in the form of verification (discussed later) and plans for the validation testing (discussed later).

SOFTWARE TESTING— MYTHS AND FACTS

Myth 6: The purpose of testing is to check the functionality of the software.

Truth: The goal of testing is to ensure quality of the software

But quality does not imply checking only the functionalities of all the modules.

SOFTWARE TESTING— MYTHS AND FACTS

Myth 7: Anyone can be a tester.

Truth: Software testing as a career also needs training for various purposes, such as to understand:

*** Various phases of software testing life cycle,**

*** Recent techniques to design test cases,**

*** Various tools and how to work on them, etc..**

SOFTWARE TESTING— MYTHS AND FACTS

Myth 1: Testing is a single phase in SDLC .

Myth 2: Testing is easy.

Myth 3: Software development is worth more than testing.

Myth 4: Complete testing is possible.

Myth 5: Testing starts after program development.

Myth 6: The purpose of testing is to check the functionality of the software.

Myth 7: Anyone can be a tester.

EVOLUTION OF SOFTWARE TESTING

The evolution of software testing was discussed in three phases namely;

(1) Software Testing 1.0:

Software testing was considered as a single phase to be performed after coding of the software in SDLC.

No test organization was there.

A few testing tools were present but their use was limited due to high cost.

Management was not concerned with testing, as there was no quality goal.

EVOLUTION OF SOFTWARE TESTING

(2) Software Testing 2.0:

Software testing gained importance in SDLC and the concept of early testing also started.

Testing was evolving in the direction of planning the test resources.

Many testing tools were also available in this phase.

EVOLUTION OF SOFTWARE TESTING

(3) Software Testing 3.0:

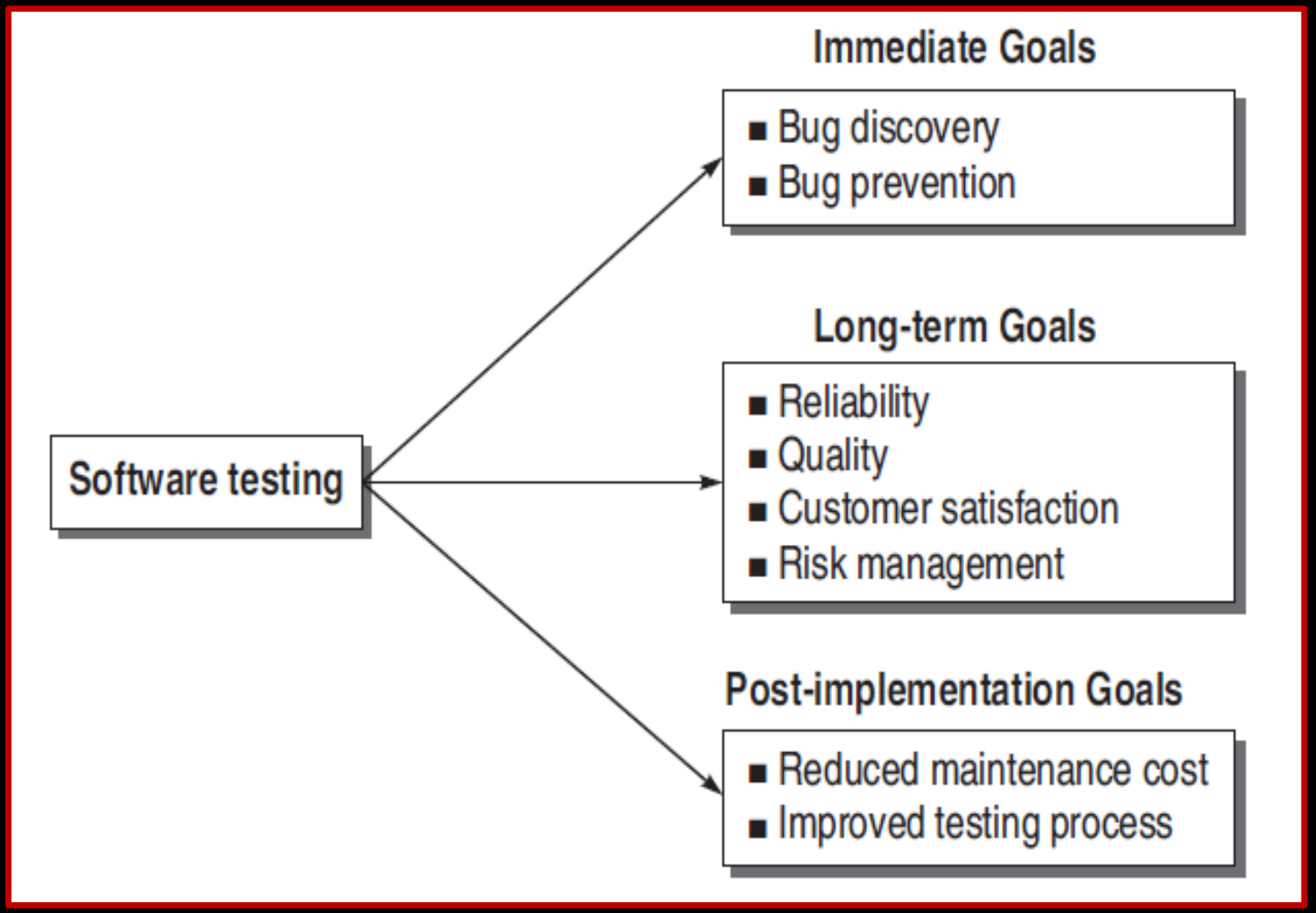
Software testing is being evolved in the form of a process (i.e. There should be a process which gives us a roadmap of the overall testing process)

All controlling and monitoring activities performed by the Test managers.

The management is actively involved in this phase.

GOALS OF SOFTWARE TESTING

The goals of software testing may be classified into three major categories as shown:



GOALS OF SOFTWARE TESTING

(1) Short-term (or) immediate goals:

These goals may be set in the individual phases of SDLC. Some of them are discussed below.

(i) Bug discovery:

More the bugs discovered at an early stage, better will be the success rate of software testing.

GOALS OF SOFTWARE TESTING

(ii) Bug prevention:

Everyone in the software development team gets to learn how to code safely such that the bugs discovered should not be repeated in later stages or future projects.

GOALS OF SOFTWARE TESTING

(2) Long-term goals:

These goals affect the product in the long run, Some of them are discussed here.

(i) Quality:

The first goal of the testing process is to enhance the quality of the software product.

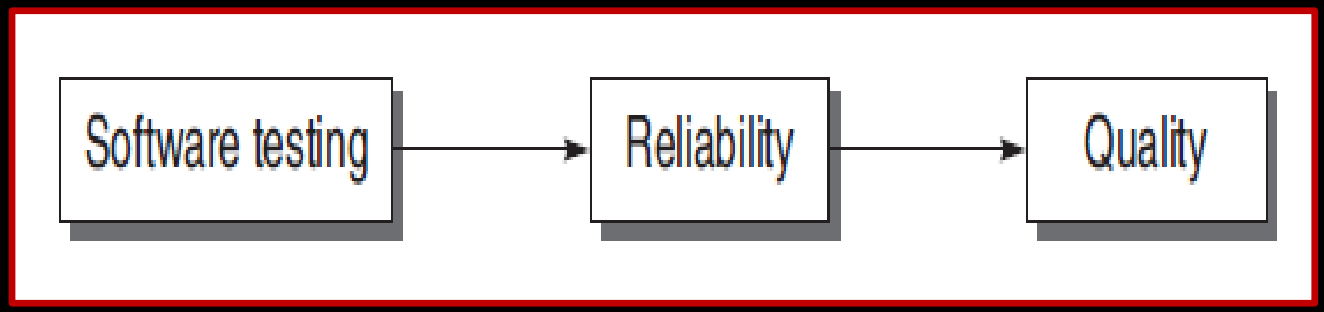
Since software is also a product, its quality is primary from the users' point of view.

Testing produces Reliability and Quality

(ii) Reliability:

Reliability is a "matter of confidence" that the software will not fail, and this level of confidence increases with rigorous testing.

The confidence in reliability, in turn, increases the quality, as shown in Figure.

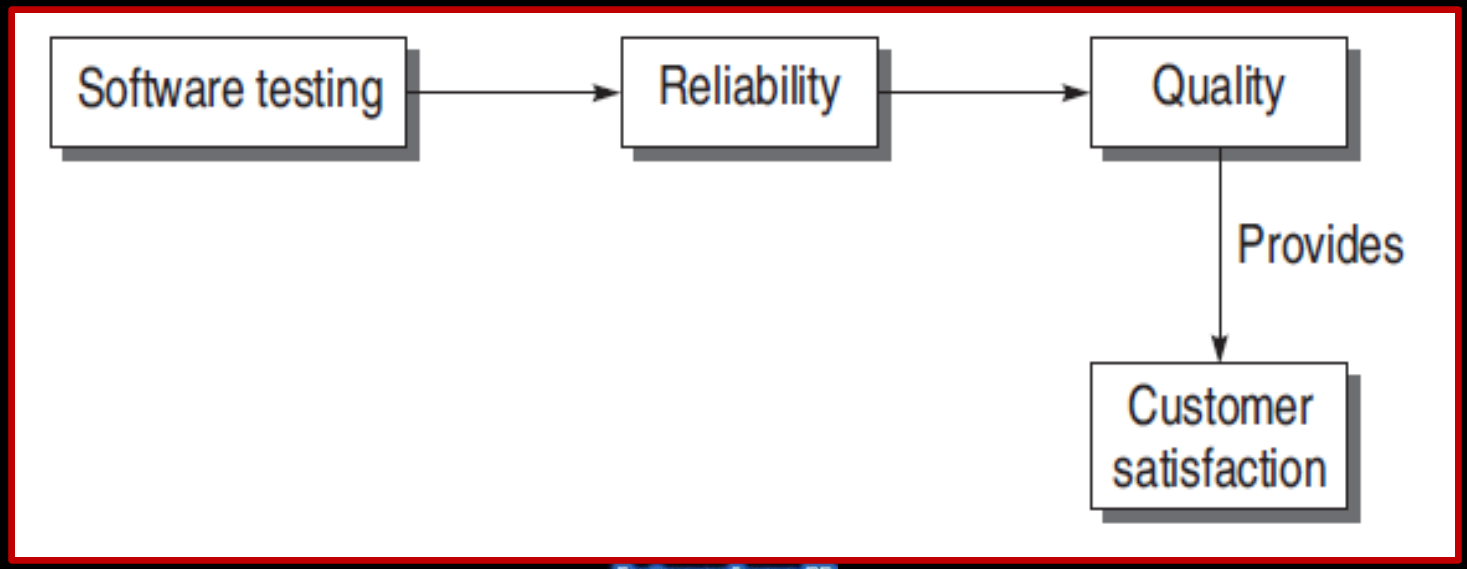


Quality leads to customer satisfaction

(iii) Customer satisfaction:

If we want the customer to be satisfied with the software product, then testing should be effective and through. (i.e. satisfy the user for all the specified requirements & unspecified requirements)

A effective testing process achieves reliability, reliability enhances the quality, and quality in turn, increases the customer satisfaction.



Testing controlled by Risk factors

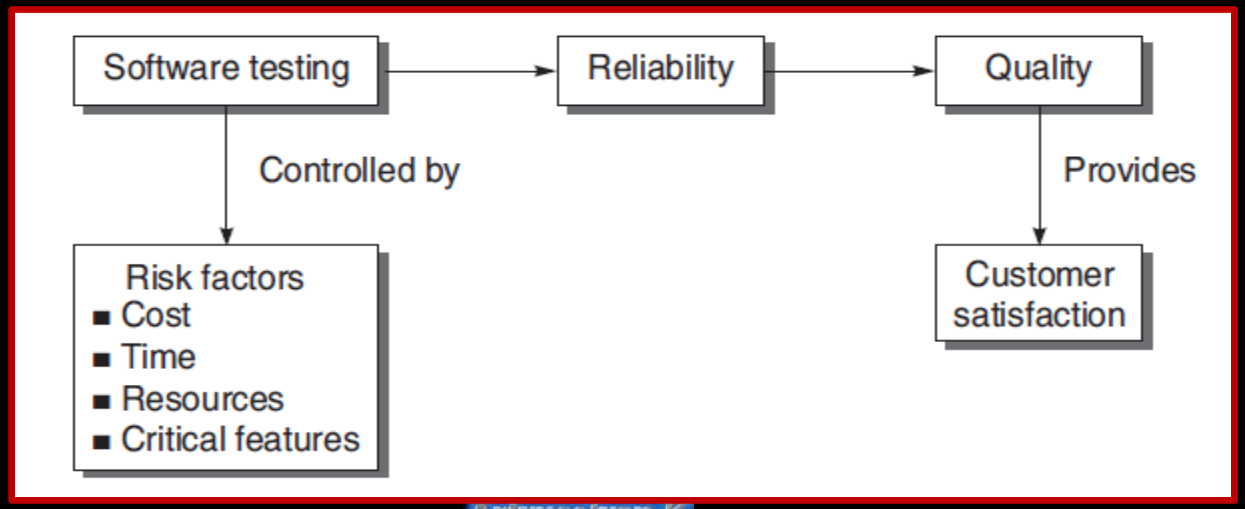
(iv) Risk management:

Risk is the probability that "undesirable events" will occur in a system.

Software testing may act as a control, (which can help in eliminating or minimizing risks.)

The testers' responsibility is to evaluate business risks (such as cost, time, resources, and critical features of the system)

Testers should also categorize the levels of risks after their assessment (like high-risk, moderate-risk, low-risk)



GOALS OF SOFTWARE TESTING

(3) Post-implementation goals:

These goals are important after the product is released. Some of them are discussed here.

Reduced maintenance cost:

The maintenance cost of any software product is not its physical cost,(as the software does not wear out.)

The only maintenance cost in a software product is its failure due to errors.

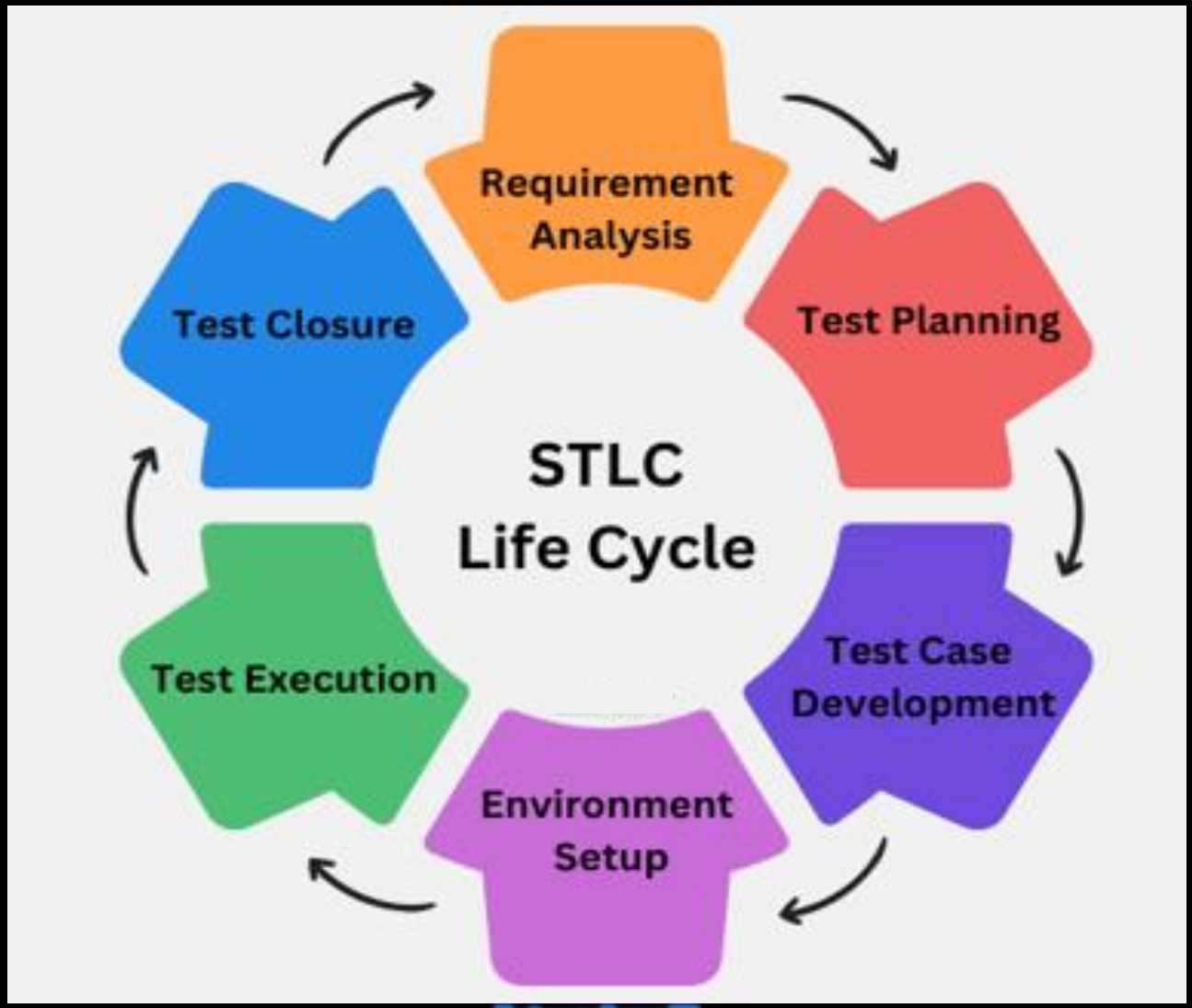
Post-release errors are costlier to fix, as they are difficult to detect.

Thus, if testing has been done rigorously and effectively, then the chances of failure are minimized. (i.e. in turn the maintenance cost is reduced.)

SOFTWARE TESTING LIFE CYCLE (STLC)

SOFTWARE TESTING LIFE CYCLE (STLC)

A series of phases that are performed by software engineers to test that the software is free of bugs and faults.



1. Requirement Analysis Phase

It involves gathering information about the software requirements, (Reviewing these requirements allows the Quality Assurance (QA) team to know exactly what needs to be tested)

Testing team studies the functional/non-functional requirements from testing point of view.

Identifying any potential risks or issues that may impact the testing process.

2. Test Planning Phase

A test plan document is made during this stage that outlines the test strategy.

This document is a blueprint of the entire testing process, which includes,

- Steps involved within it,
- Tools required and
- Other detail that will be important for testing.

3. Test Case Development Phase

Test case are detailed test scenarios that are executed to check every functionality of the software product.

For example, to test that the login function of the app they will perform the following steps:

This scenario can result in a number of test cases, for example:

Test Case # 1: Test result when correct email address and password is entered
Test Case # 2: Test result when incorrect email address and password is entered
Test Case # 3: Test result when correct email address and incorrect password is entered
Test Case # 4: Test result when incorrect email address and correct password is entered
Test Case # 5: Test result when email and password are left blank

For a thorough testing process, multiple detailed test cases will be written for each functionality so any error or bug can be identified and fixed.

4. Test Environment Setup Phase

It is important to ensure that the environment in which the software is being tested closely matches the environment in which the application will be used after deployment.

This is independent activity and can be started along with test case development.

5. Test Execution Phase

This is the stage where all the test cases that were developed are executed.

There are two ways of execution: Manual and Automatic.

In manual testing, a Q/A engineer manually performs the test cases and records the results.

While for automated testing, test scripts are developed that are automatically run in an automated testing tool to check the results of each test case.

There are 3 possible results of test case execution: Passed, Failed, or Blocked.

Passed: A test case is passed when it is executed and the result is as per the required output. For example, in our login functionality example, when a correct email address and password is entered and the user is successfully logged in, it shows that the test case is passed. Or in other words, no bug is identified.

Failed: If an incorrect email and password was entered and the user was still able to log in, instead of being shown an error, it shows that the test case failed.

Blocked: A blocked test case is when a case execution fails due to some internal or external defects in the application.

For any test cases that are failed or blocked, they are reported back to the development team who fix the bug. Once it is fixed, it is retested to ensure that the functionality performs perfectly.

6. Test Closure Phase

Test closure report is created by a QA engineer and is reviewed by all the stakeholders.

Test closure reports outlines the final status of a testing project.

Test Closure Report

Table of Contents

1. Document information

Project start date: Jan 2025

Project end date: March 2026

2.Project Overview

Project type	Testing
Project duration	Jan 2025 to March 2026
Operating system	Windows
Browsers tested	Chrome, IE, Mozilla, Safari
Testing performed	Desktop
Mode of testing	Manual
All reported issues have been solved	Yes
All defects logged	Yes
Critical issues	No
Test closure activities completed and signed off	Yes

3. Tests Executed

Test cycle number	Planned number of tests per cycle	Total number of tests executed	Total number of passed test cases	Passed %	Number of failed test cases	Failure %
10	10	118	118	100	0	0

4. Test Completion Matrix

Cases	Result
Total number of test cases	118
Number of test cases executed	118
Number of test cases passed	118
Number of test cases failed	0
Number of test cases pending	0
Number of test cases in progress	0
Number of test cases blocked	0
Number of test cases rejected	0
Number of test cases deferred	0
Number of active defects	0

5. Defect Metrics: Used to find out the current status of the defects linked to the test cases.

Showstopper defect: A bug which stops/restricts the testing to move ahead with that specific functionality.

Priority	Number of defects opened	Number of defects closed	Number of defects on hold	Number of defects rejected	Defects open at the end of the test phase
Showstopper	5	5	0	0	0
Critical	11	11	0	0	0
High	15	15	0	0	0
Medium	5	5	0	0	0
Low	45	0	0	0	0

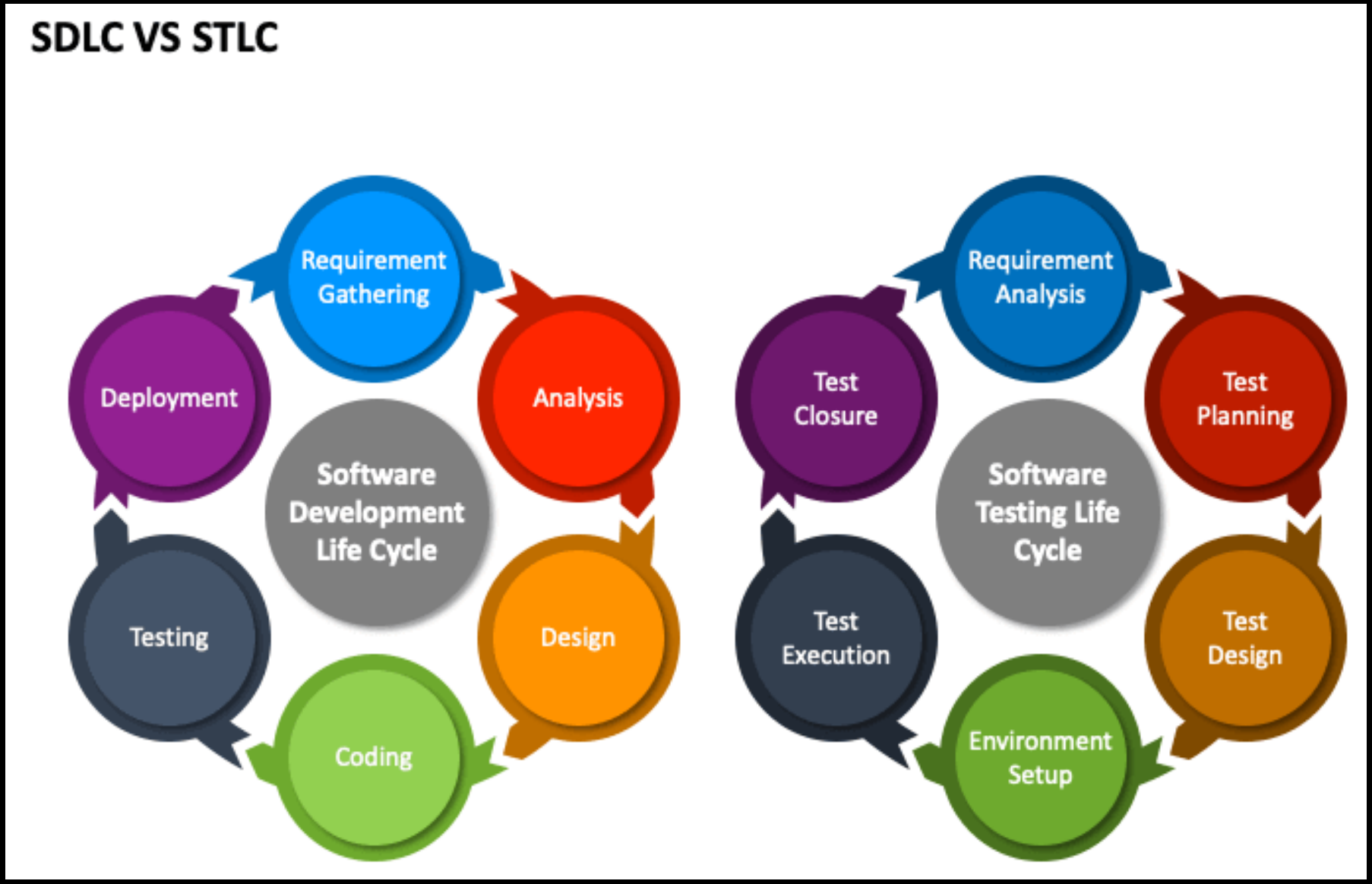
6. Exit Criteria Status

Exit criteria	Criteria met (Yes/No)	If No, describe action plan
All test cases have been successfully executed.	Yes	
All expected and actual results are captured and documented with the test cases.	Yes	
All identified critical or high severity defects have been fixed and retested.	Yes	
Any unresolved defects are documented and signed-off by the project manager.	Yes	
All defects logged	Yes	
Pass rate is over	Yes	

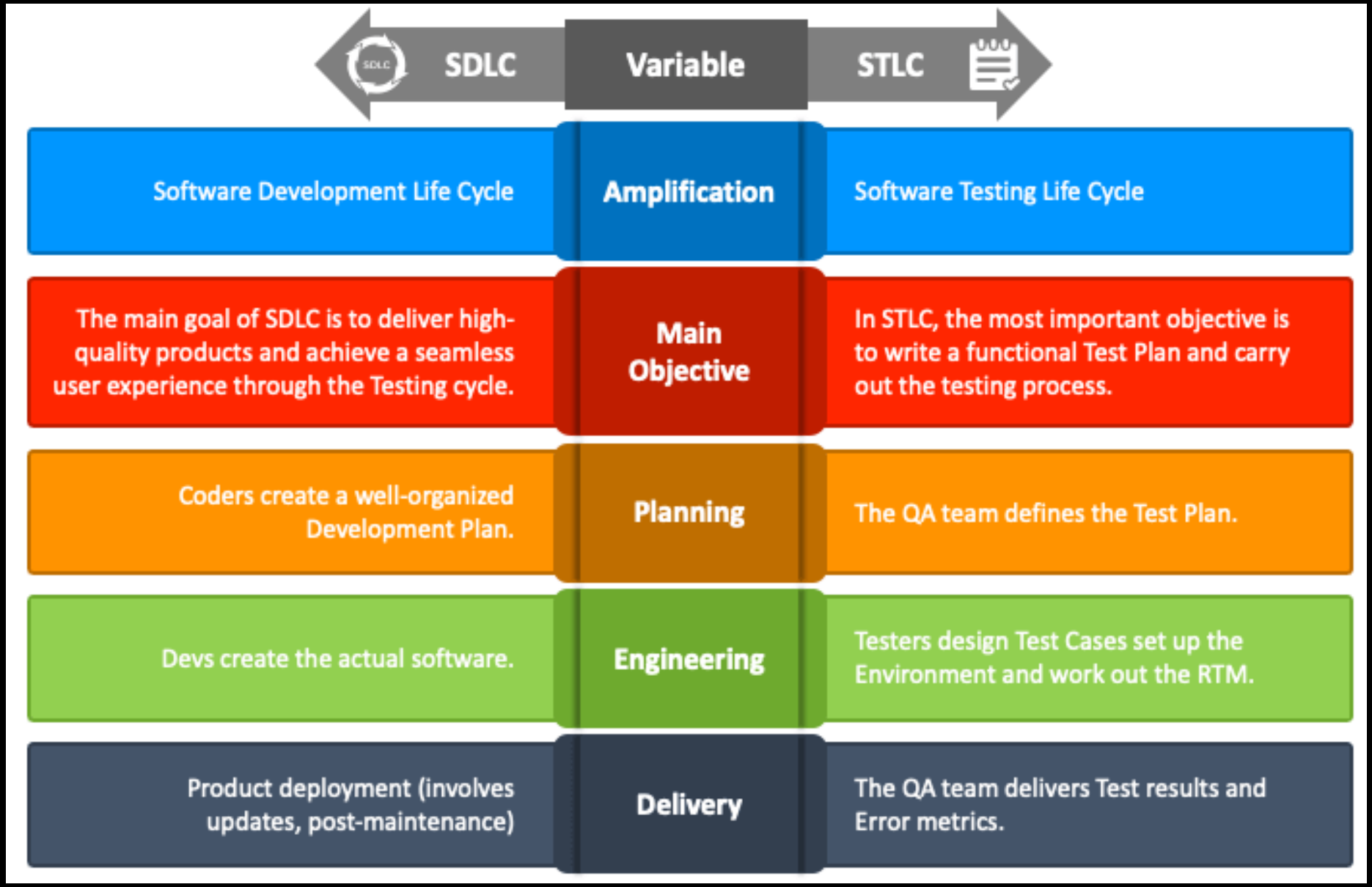
7. Approvals: The test closure approval needs to be provided by the following:

Role	Name	Signature
Business owner		
Project manager		
QA lead		

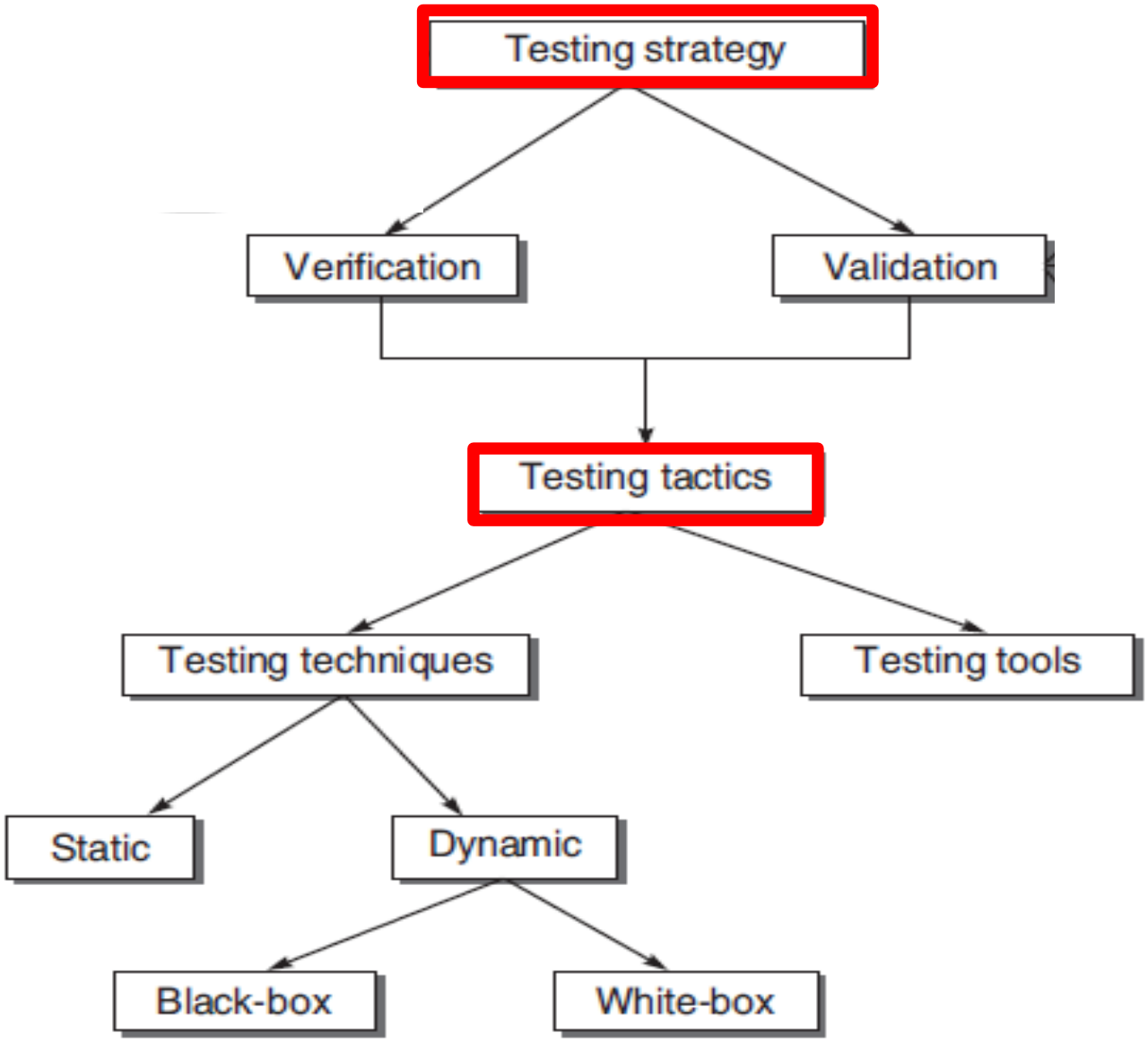
STLC Vs. SDLC: How Do They Differ?



STLC Vs. SDLC: How Do They Differ?



SOFTWARE TESTING METHODOLOGY



1. TESTING SRATEGY

Testing strategy is the planning of the whole testing process into a well-planned series of steps.

Strategy provides a roadmap that includes very specific activities that must be performed by the test team in order to achieve a specific goal.

Test Strategy Matrix:

This matrix becomes an input to develop the testing strategy.

The matrix is prepared using test factors and test phase.

Test Factors	Test Phase					
	Requirements	Design	Code	Unit test	Integration test	System test

Creating a test strategy: (Example)

Suppose a new operating system has to be designed, which needs a test strategy. The following steps are used:

Test Factors	Test Phase					
	Requirements	Design	Code	Unit test	Integration test	System test
Portability	Is portability feature mentioned in specifi-cations according to different hardware?					Is system testing performed on MIPS and INTEL platforms?
Service Level	Is time frame for boot-ing mentioned?	Is time frame incor-porated in design of the module?				

Verification:

It refers to the set of activities that ensures correct implementation of functions in a software.

Verification can be applied to all stages of SDLC.

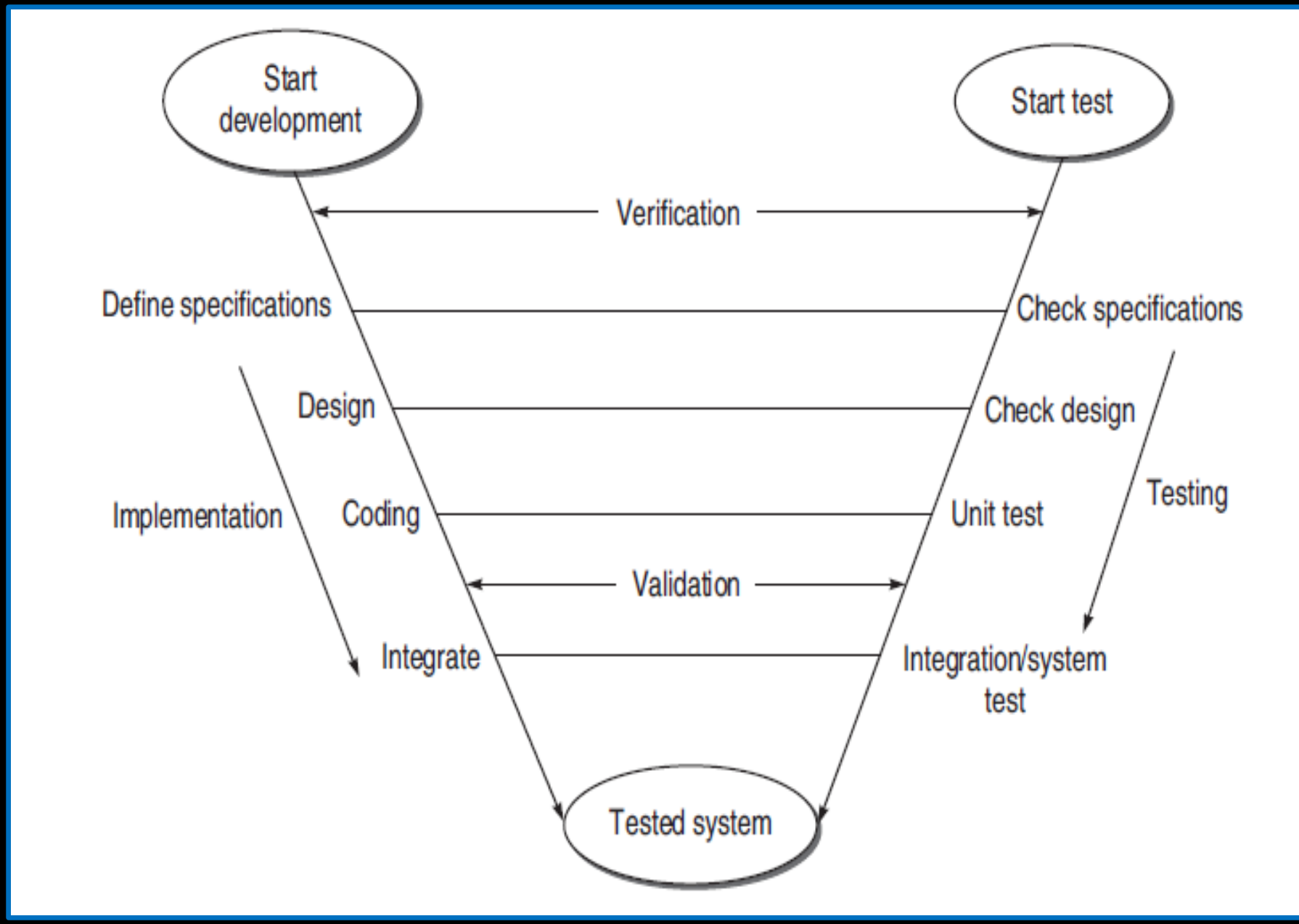
Validation:

It is a very general term to test the software as a whole in conformance with customer expectations.

The validation process starts in the later stages of SDLC.

Verification Process	Validation Process
It means **"Are we implementing the software right?"**	It means **"Are we implemented the right software?"**
Developer developed: ✓ Chatting Functionality then Verify it. ✓ Status Functionality then Verify it. ✓ Audio Call Functionality then Verify it. ✓ Video Call Functionality then Verify it. ✓ Share Photos Functionality then Verify it.	Tester Validate complete developed product at same time. ✓ Validate Chatting, Status, Audio Call, Video Call, Share Photos functionality at a same time.

V-Testing Model



2. TESTING TACTICS

(i). Unit Testing

Unit testing, also called component testing, is the most basic type.

It aims to ensure that each unit of code,(such as a function or method) conforms to the requirements of the program specification and operates as intended.

Benefits of unit testing

Early bug detection

Unit tests can help catch bugs and defects early in the development cycle before they become more difficult and expensive to fix.

Improved code quality

Writing unit tests forces developers to think carefully about the code they are writing and ensures that each unit of code is working as expected.

Faster feedback

Unit tests can provide developers with fast and immediate feedback on their code changes, helping to reduce the time and effort required for debugging and troubleshooting.

Easier maintenance

Unit tests make it easier to refactor and modify code without introducing new bugs or breaking existing functionality.

Unit Test Case (Example)

Sample Test case for logging into the e-commerce application.

Since it is a Unit Test case, we can take the individual elements like the 'Username' and 'Password' text boxes and the 'Submit' button.

Test Case 1: Username Input Text box

Step 1: The user should be able to click on the 'Username' text box and see the cursor inside

Step 2: 'Username' text box should accept alphabets.

Step 3: 'Username' text box should accept numbers.

Step 4: 'Username' text box should have at least one alphabet

Step 5: 'Username' text box should have at least one number

Step 6: 'Username' text box should not accept special characters.

Step 7: 'Username' text length should not exceed 15 alphanumeric characters.

Test Case 2: Password Input Text box

Step 1: The user should be able to click on the 'Password text box and see the cursor inside

Step 2: 'Password' text box should accept alphabets.

Step 3: 'Password' text box should accept numbers.

Step 4: 'Password' text box should have at least one alphabet

Step 5: 'Password' text box should have at least one number

Step 6: 'Password' text box should have at least one special character.

Step 7: 'Password' text length should be eight and above.

Test Case 3: Submit button

Step 1: The user should be able to click on the 'Submit' button,

Step 2: 'Username' text box should not be empty.

Step 3: 'Password' text box should not be empty.

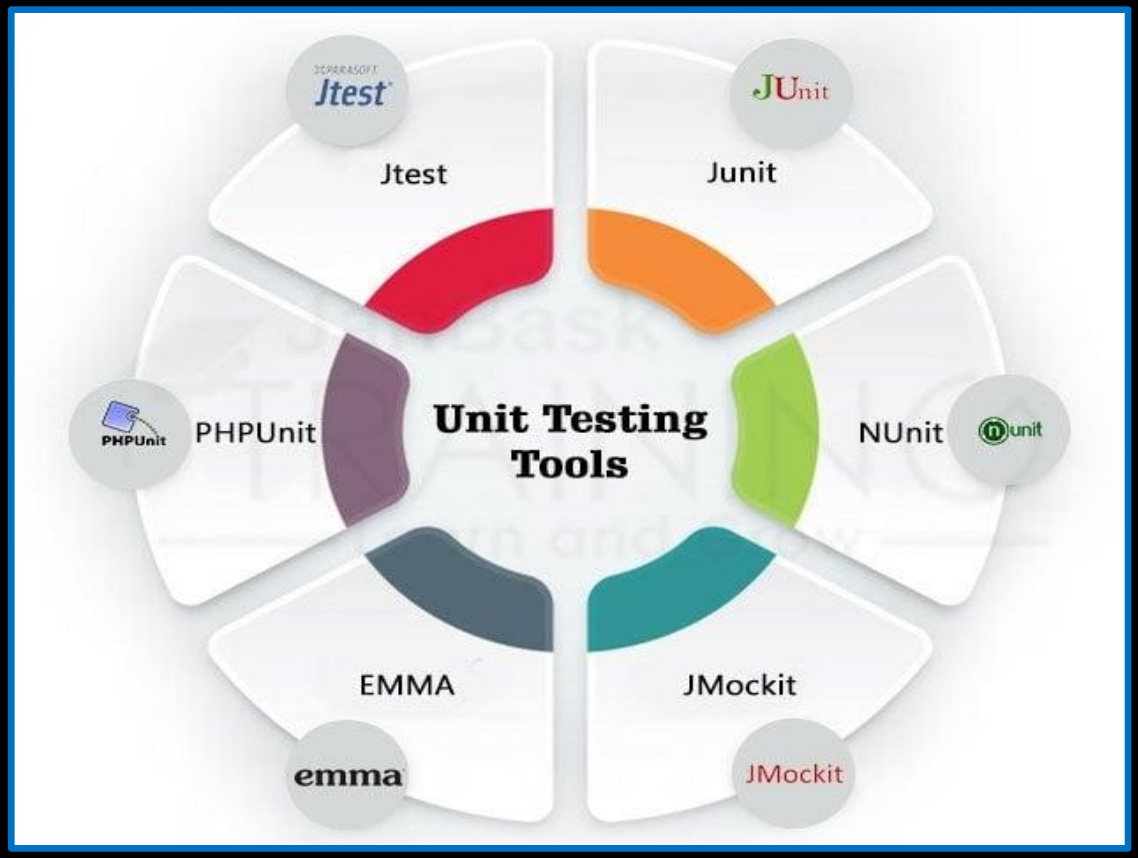
Step 4: Validate the Username with the existing database.

Step 5: Validate Password with the existing database.

Step 6: If the Username and Password validation are successful, the user should be taken to the 'Home' page.

Step 7: If the Username and Password validation are unsuccessful, the error message 'Invalid Credentials' should be displayed.

Tools Used for Unit Testing



TestNG

(ii). Integration Testing

The main purpose of integration testing is to validate that different software components, subsystems, or applications work together as a system to achieve the desired functionality and performance.

Integration Test Case (Example)

Sample test case for retail website:

Objective: To test the integration of the shopping cart, payment gateway, and order management systems.

Prerequisites:

- A customer account must be created
- The customer must have at least one product in their shopping cart

Steps:

1. Log in to the retail website as a customer
2. Navigate to the shopping cart and verify that the correct products are listed
3. Select a payment method and enter payment information
4. Click the “Place Order” button
5. Verify that the payment is processed successfully
6. Verify that the order is listed in the order management system
7. Verify that the order details, such as the products, shipping address, and payment method, are correct

Integration Test Case (Example)

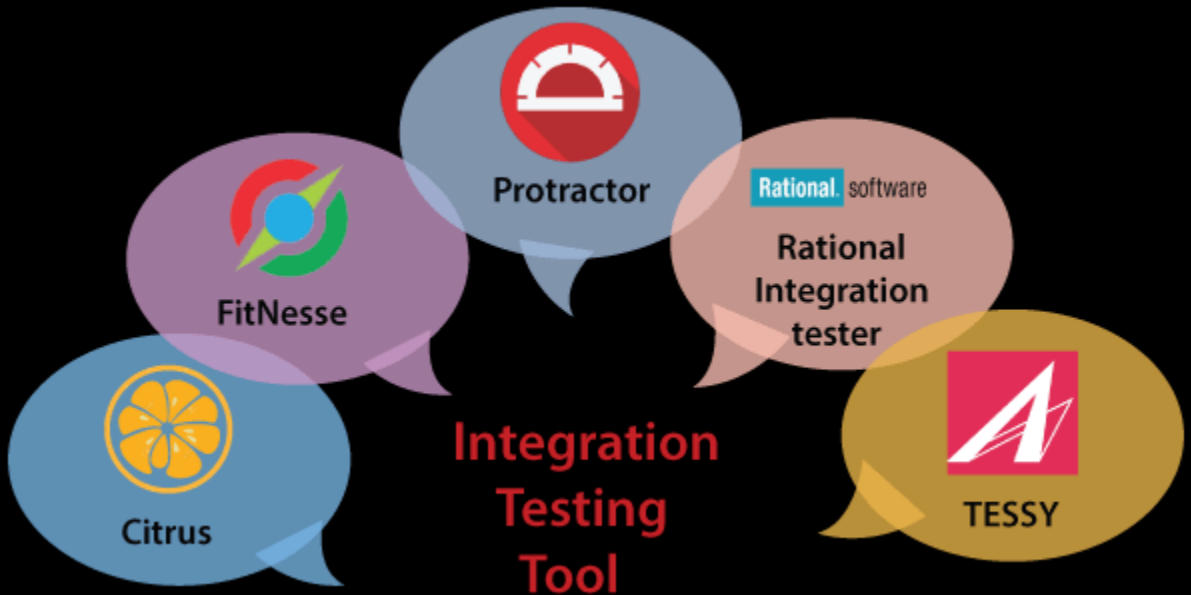
Expected Results:

- The payment should be processed successfully
- The order should be listed in the order management system
- The order details should be correct

Tools for Integration Testing

Jenkins

Selenium



(iii). System Testing

System testing is also known as black-box testing because it focuses on the external parts of the system.

It takes place after integration testing and before the acceptance testing.

System Testing Examples

- **Software Applications:** Use cases for an online airline's booking system – customers browse flight schedules and prices, select dates and times, etc.
- **Web Applications:** An e-commerce company lets you search and filter items, select an item, add it to the cart, purchase it, and more.
- **Mobile Applications:** An UPI app let you do mobile recharge or transfer money securely. So, first, you have to select the mobile number, then the biller name, recharge amount, and payment method, and proceed to pay.
- **Games:** For a gaming app, check the animation, landscape-portrait orientation, background music, sound on/off, score, leaderboard, etc.
- **Operating Systems:** Login to the system with your password, check your files, folders, apps are well placed and working, battery percentage, time-zone, go to the 'settings' for additional checkups, etc.
- **Hardware:** Test the mechanical parts – speed, temperature, etc., electronic parts – voltage, currents, power input-output, communication parts- bandwidths, etc.

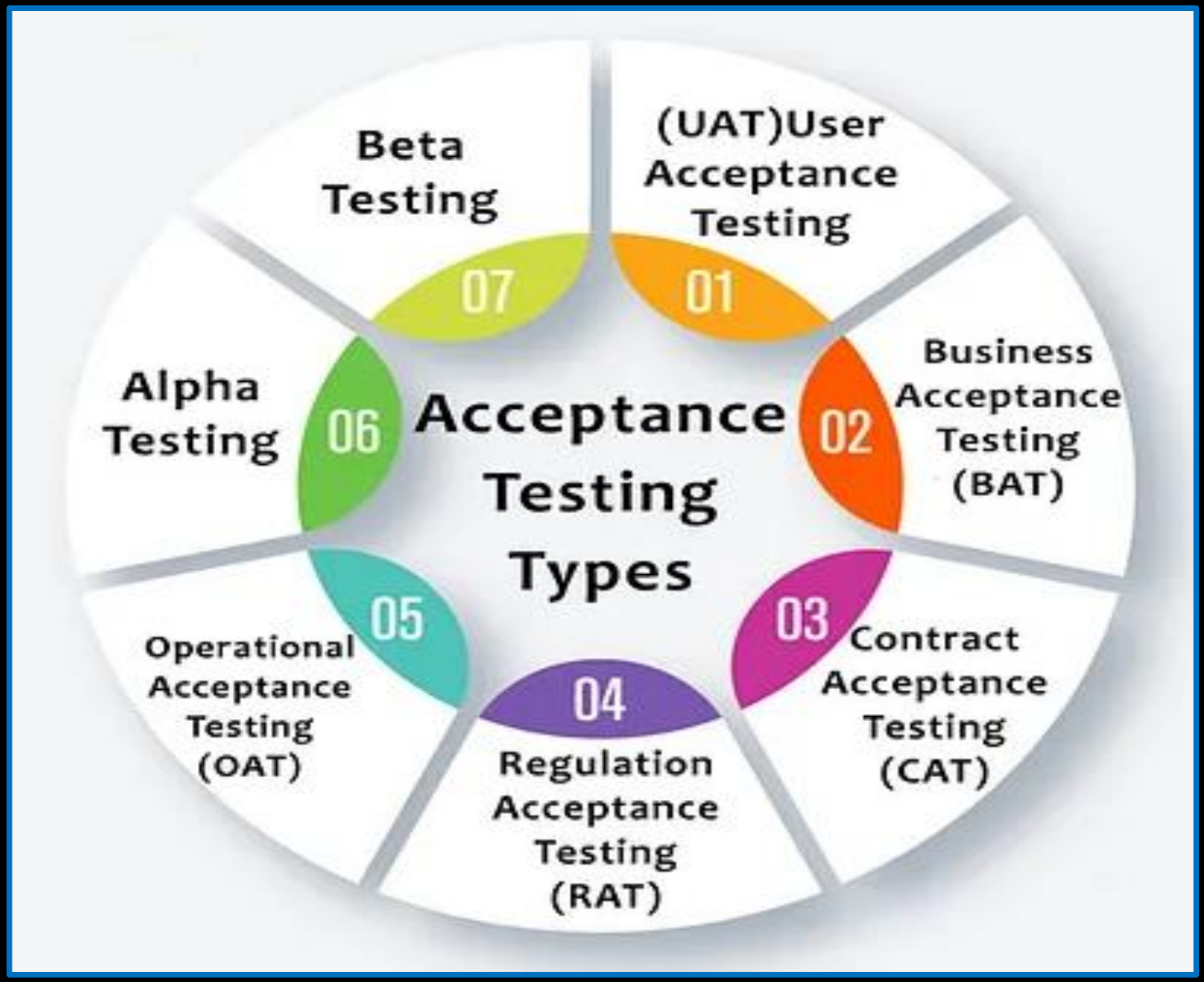
Tools for System Testing

(iv). Acceptance Testing

The functionality is verified to ensure the software product meets with the specifications agreed with the client.

It's the last phase of the software testing process, and it's important before making the software available for actual use.

Acceptance Testing- Types



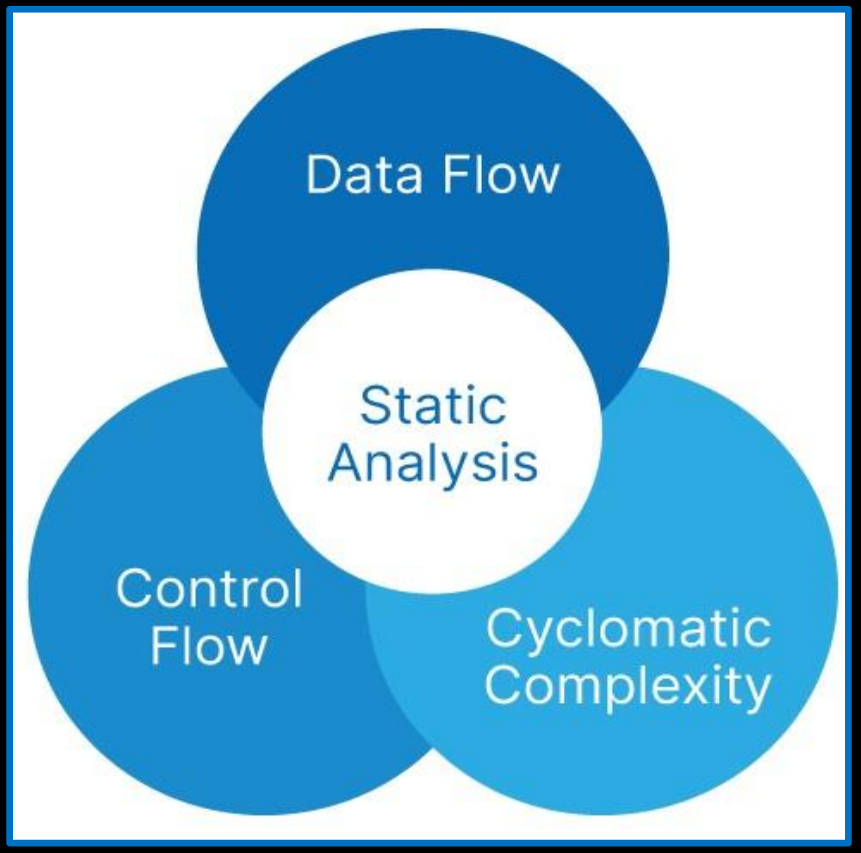
Tools for Acceptance Testing

(2.1). Static Testing

Testing is carried out using associated documents, such as design documents, user documents, webpage content, etc., but the application's code is not executed.

This is performed at the early stage of development to identify the issues in the project documents in multiple ways, (namely reviews, walkthroughs, and inspections).

Types of Static Testing

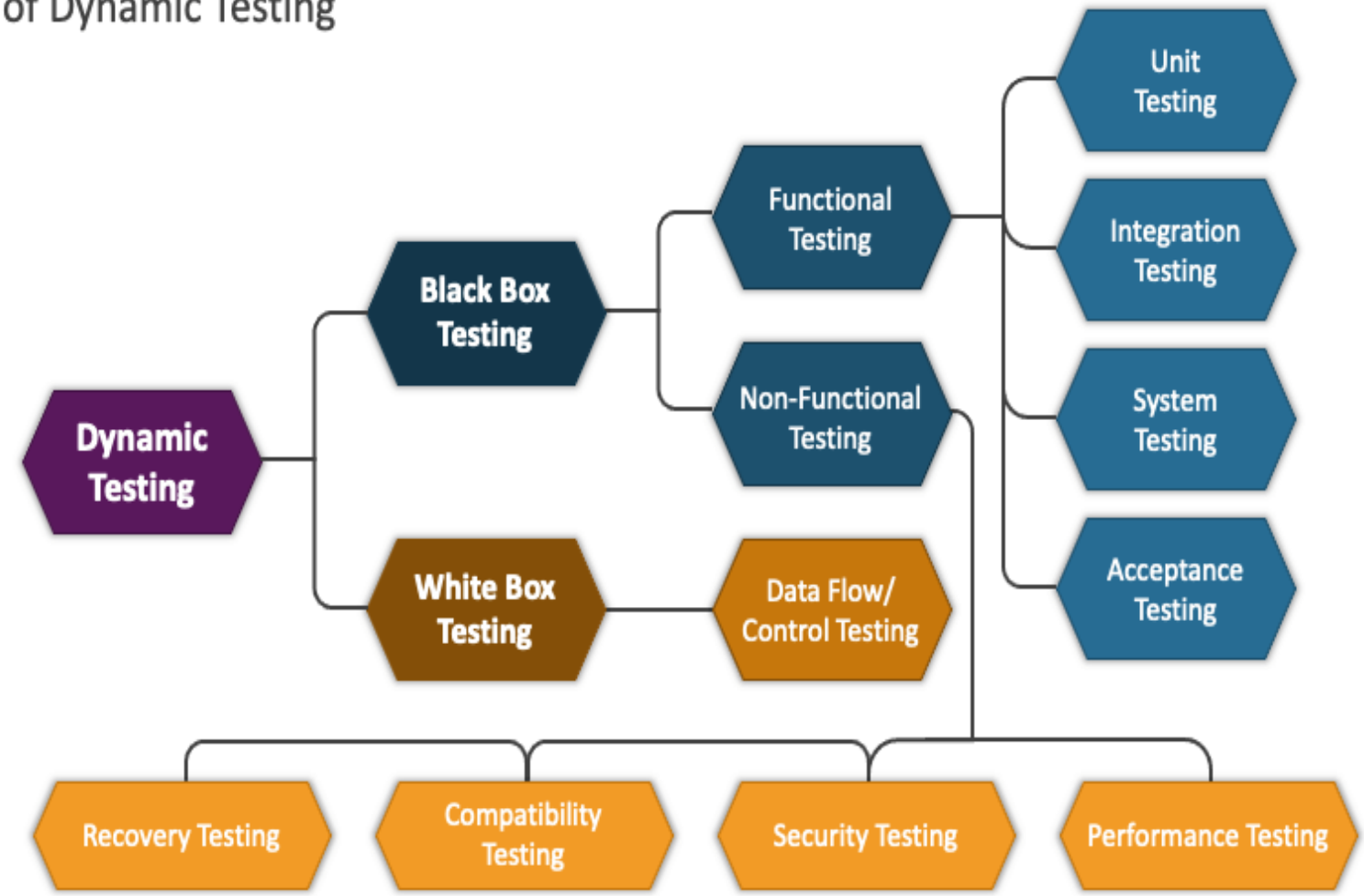


Tools for Static Testing

(2.2). Dynamic Testing

DYNAMIC TESTING

Types of Dynamic Testing



(i) Black Box Testing

Engineers have to test the software as per the requirements and specifications.

They don't need to know about the internal implementation or coding of the software. So, programming knowledge is not necessary for this testing.

Black box testing is performed after white box testing.

(ii) White Box Testing

It **mandates the coding knowledge of a tester**, because it needs to **test internal coding implementation** and algorithms for the system.

For this testing, tester have to **execute a programming line-by-line** to find **whether there are errors** in the line.