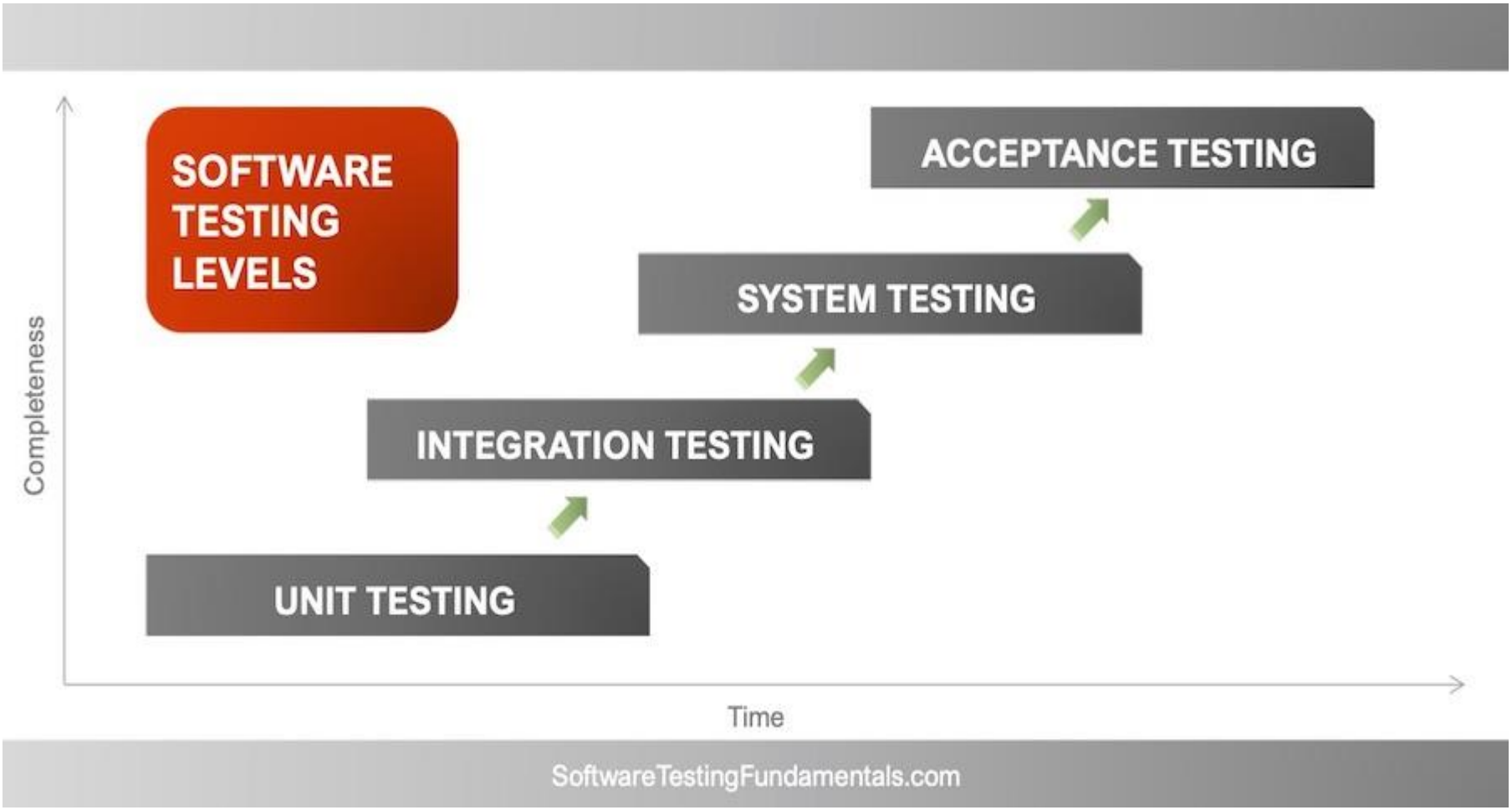
CHƯƠNG 3
CÁC CẤP ĐỘ KIỂM THỬ PHẦN MỀM

09/2022

NỘI DUNG

- **Software Testing Levels**
- **Unit Testing**
- **Integration Testing**
- **System Testing**
- **Acceptance Testing**

Software Testing Levels



Unit Testing

- What is Unit Testing?
- Why Unit Testing is important to perform?
- What are the benefits of Unit Testing?
- What are the types of Unit Testing?
- Who performs Unit Testing?
- How to do Unit Testing?
- Design Testcase

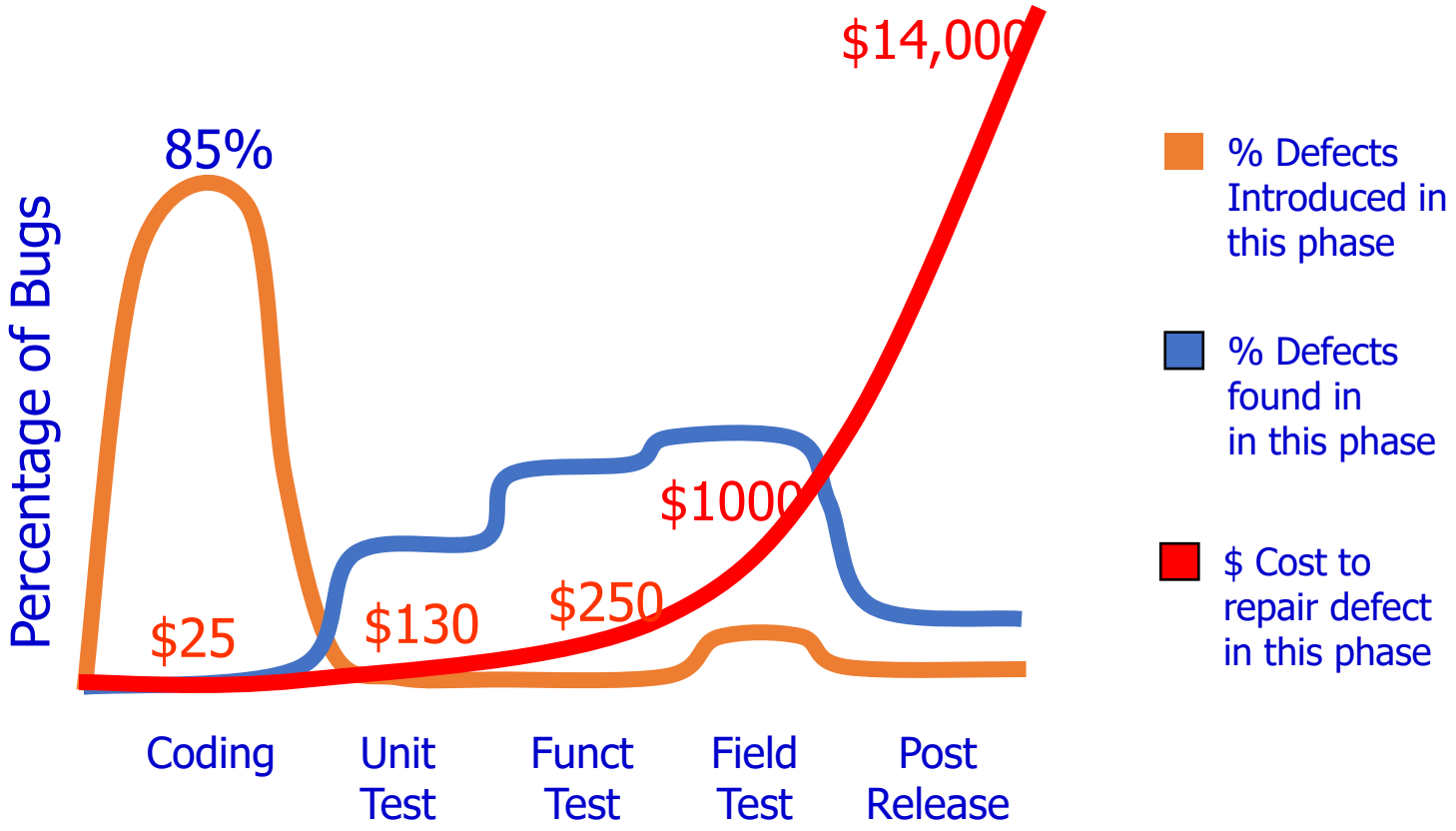
What is Unit Testing?

- Kiểm thử module (hay kiểm thử đơn vị) là quá trình kiểm thử từng chương trình con, từng thủ tục nhỏ trong chương trình.
- Một số động cơ của việc kiểm thử đơn vị:
  - Kiểm thử đơn vị là 1 cách quản lý nhiều phần tử cần kiểm thử, bắt đầu tập trung chú ý trên từng phần tử nhỏ của chương trình.
  - Kiểm thử đơn vị giúp dễ dàng việc debug chương trình.
  - Kiểm thử đơn vị tạo cơ hội tốt nhất cho việc thực hiện kiểm thử đồng thời bởi nhiều người.

What is Unit Testing?

- Mục đích của kiểm thử đơn vị: so sánh chức năng thực tế của từng module với đặc tả chức năng hay đặc tả interface của module đó.
- Sự so sánh này có tính chất :
  - Không chỉ ra việc module có thoả mãn đầy đủ đặc tả chức năng?
  - Mà chỉ ra việc module có làm điều khác biệt gì so với đặc tả của module.

Cost of bugs

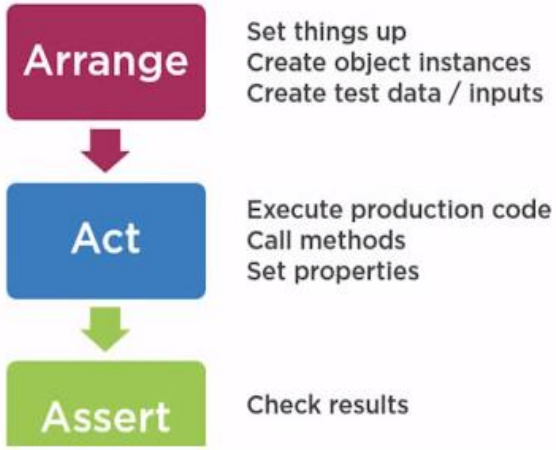


Source: *Applied Software Measurement*, Capers Jones, 1996

What is Unit Testing?

- A typical unit test consists of three phases:
  - **Arrange:** initializes a small piece of an application it wants to test
  - **Act:** is the addition phase where it adds a stimulus to the system under test and finally
  - **Assert:** is the result phase where it observes the resulting application behavior

```
@Test
public void test_login_succeeds_if_input_correct_user_and_password() {
    // arrange
    String user = "abc";
    String pass = "abc";
    // act
    String result = login(user, pass);
    // assert
    assertEquals( expected: "Login ok", result);
}
```



What are the types of Unit Testing?

- **White Box Testing (**transparent testing, glass box testing): the functional behavior of the software is tested by the developers in order to validate their execution.
- **Black Box Testing:** the unit testers are allowed for testing the structure of the function, i.e. the user interface, input, and output.
- **Gray Box Testing:** the tester is partially aware of the system functionality and uses this approach for testing various test methods, for executing test suites, and to perform the risk assessment.

How to do Unit Testing?

The workflow of unit testing is performed in 4 stages:

1. Creating test cases
2. Reviewing test cases
3. Baselining test cases
4. Executing test cases

How to do Unit Testing?

Unit Testing process involves:

The developers write the code in the application for testing the function and would wait for the application to get deployed for removing the test code

The code is then isolated by the developers to validate the dependencies between the code and other units → identifying and eliminating the dependencies.

Developers significantly make use of Unit test frameworks or unit testing tools for developing automated test cases.

While executing the test cases, the unit test frameworks help to flag and report the failed test cases. Also, based on the failures in the test cases, the unit test frameworks help to stop the related testing.

Design Testcase

- Hai tài nguyên thiết yếu cần thiết cho việc thiết kế các testcase:
  - Đặc tả chức năng module: nêu rõ các thông số đầu vào, đầu ra và các chức năng cụ thể chi tiết của module.
  - Mã nguồn của module.
- Tính chất các testcase là dựa chủ yếu vào kỹ thuật kiểm thử hộp trắng:
  - Khi số lượng phần tử kiểm thử ngày càng lớn thì kỹ thuật kiểm thử hộp trắng ít khả thi hơn.
  - Việc kiểm thử sau đó thường hướng đến việc tìm ra các kiểu lỗi (lỗi phân tích, lỗi nắm bắt yêu cầu phần mềm)

Design Testcase

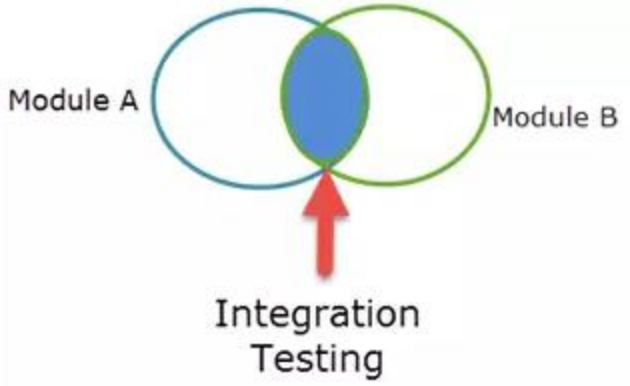
- Thủ tục thiết kế testcase
  - Phân tích luận lý của module dựa vào 1 trong các kỹ thuật kiểm thử hộp trắng.
  - Áp dụng các kỹ thuật kiểm thử hộp đen vào đặc tả của module để bổ sung thêm các testcase khác.
- Có 2 phương án để kiểm thử các module:
  - Kiểm thử không tăng tiến hay kiểm thử độc lập (Non-incremental testing)
  - Kiểm thử tăng tiến (Incremental testing)

Quy trình kiểm thử các Module

- Làm sao thiết kế được 1 tập các testcase hiệu quả.
- Cách thức và thứ tự tích hợp các module lại để tạo ra phần mềm chức năng :
  - Viết testcase cho module nào?
  - Dùng loại tiện ích nào cho kiểm thử?
  - Coding và kiểm thử các module theo thứ tự nào?
  - Chi phí tạo ra các testcase?
  - Chi phí debug để tìm và sửa lỗi?

Integration Testing

- What is Integration Testing?
- Example of Integration Test Case
- Approaches, Strategies, Methodologies of Integration Testing
- Big Bang Approach, Incremental Approach
- What is Stub and Driver?
- Bottom-up Integration, Top-down Integration:
- Hybrid/ Sandwich Integration
- How to do Integration Testing?
- Brief Description of Integration Test Plans
- Best Practices for Integration Testing



What is Integration Testing?

- Là một loại kiểm thử trong đó các module phần mềm được tích hợp một cách hợp lý và được thử nghiệm dưới dạng một nhóm.
- Mục đích: kiểm tra để lộ ra các khiếm khuyết trong tương tác giữa các module phần mềm khi chúng được tích hợp với nhau.
- Kiểm thử tích hợp tập trung vào kiểm tra giao tiếp dữ liệu giữa các module.
- Nó cũng được gọi là 'I & T' (**Integration and Testing**) **‘String Testing’, ‘Thread Testing’ .**

Example of Integration Test Case

Scenario: Ứng dụng có 3 modules: 'Login Page', 'Mailbox' and 'Delete emails' và mỗi module được tích hợp 1 cách hợp lý.

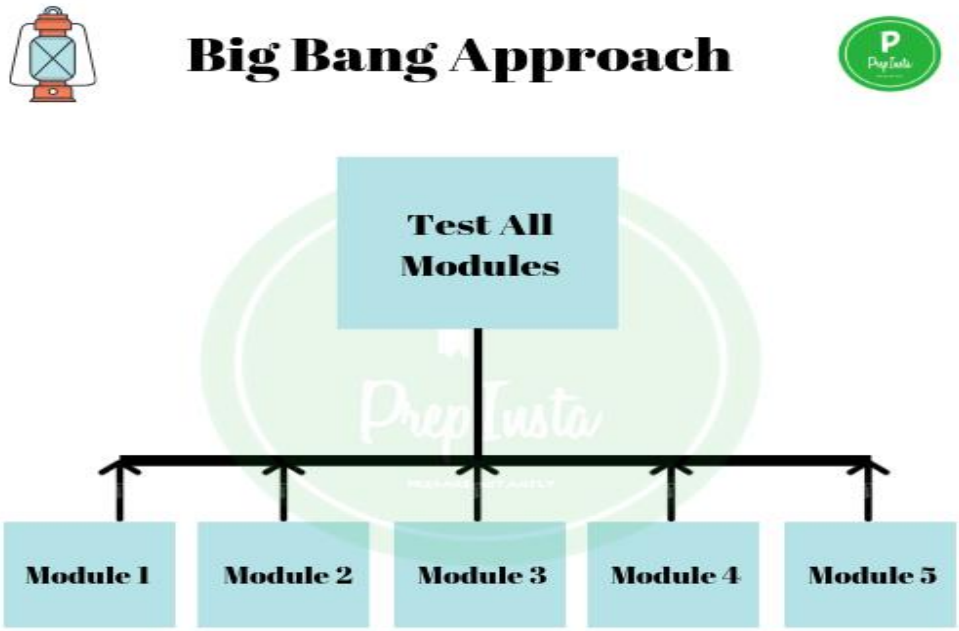
Test case ID	Test Case Objective	Test Case Description	Expected Result
1	Check the interface link between the Login and Mailbox module	Enter login credentials and click on the Login button	To be directed to the Mail Box
2	Check the interface link between the Mailbox and Delete Mails Module	From Mailbox select the email and click a delete button	Selected email should appear in the Deleted/Trash folder

Approaches, Strategies, Methodologies

- Big Bang Approach: Non-incremental testing
- Incremental Approach:
  - Top Down Approach
  - Bottom Up Approach
  - Sandwich Approach

Big Bang Approach

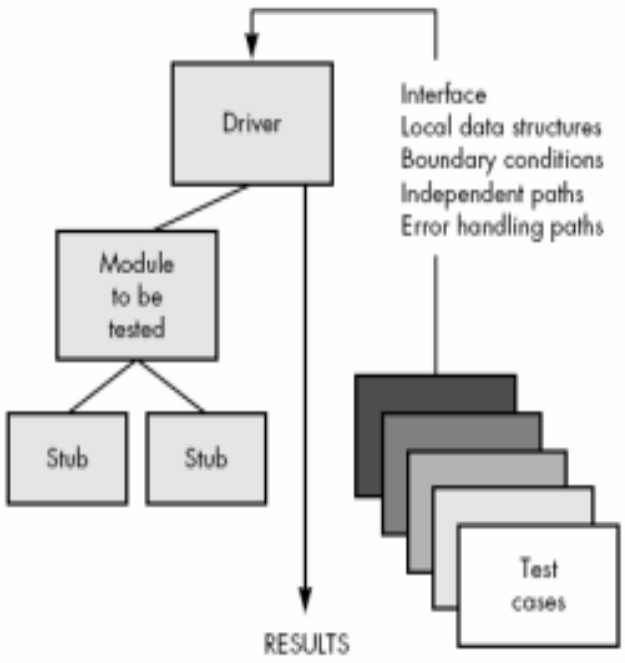
- **Big Bang Testing** is an Integration testing approach in which all the components or modules are integrated together at once and then tested as a unit.



Incremental Approach

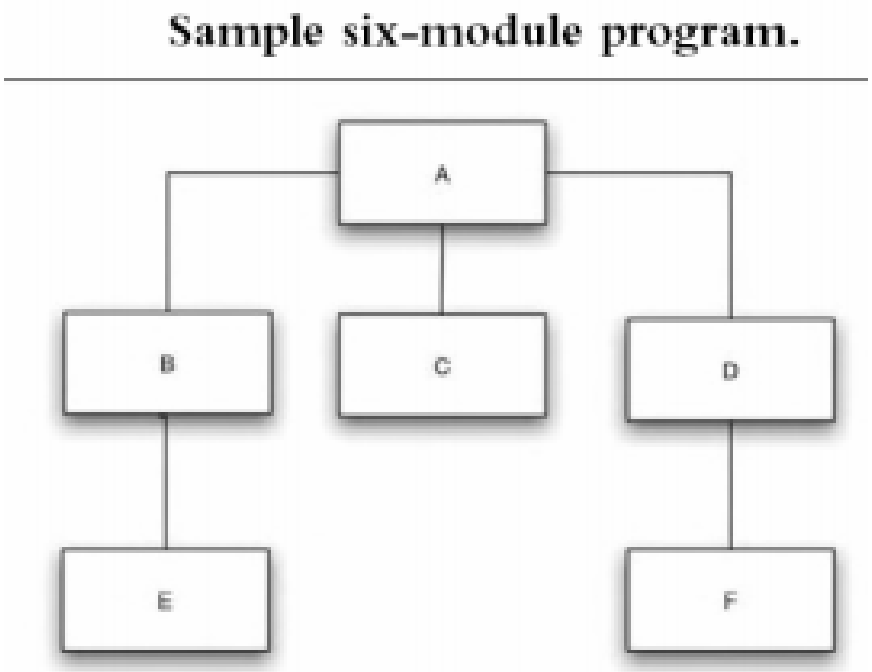
- Testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application.
- Then the other related modules are integrated incrementally and the process continues until all the logically related modules are integrated and tested successfully.
- 2 different Methods:
  - Bottom Up
  - Top Down

What is Stub and Driver?



- **Stubs and Drivers** are the dummy programs in Integration testing used to facilitate the software testing activity.
- These programs act as a substitutes for the missing models in the testing.
- They do not implement the entire programming logic of the software module but they simulate data communication with the calling module while testing.

What is Stub and Driver?

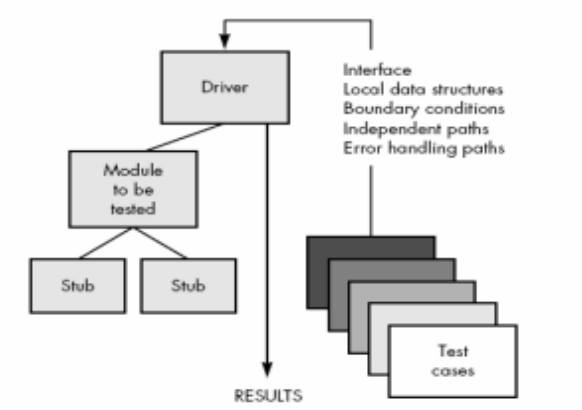


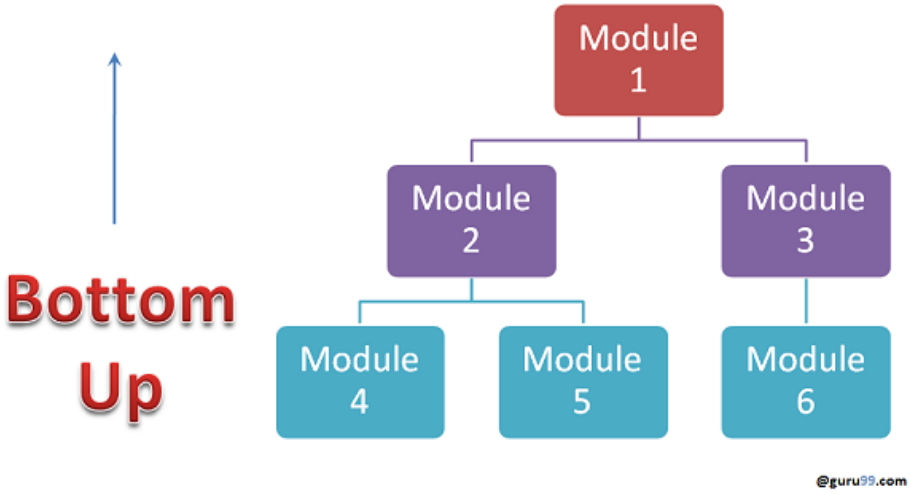
Driver là module có nhiệm vụ kích hoạt các testcase để kiểm thử module đang cần kiểm thử.

Stub là một hiện thực ở mức độ tối thiểu nào đó cho 1 module chức năng được dùng bởi module đang cần kiểm thử.

Bottom-up Integration Testing

- **Bottom-up Integration Testing** is a strategy in which the lower level modules are tested first.
- These tested modules are then further used to facilitate the testing of higher level modules.
- The process continues until all modules at top level are tested.
- Once the lower level modules are tested and integrated, then the next level of modules are formed.





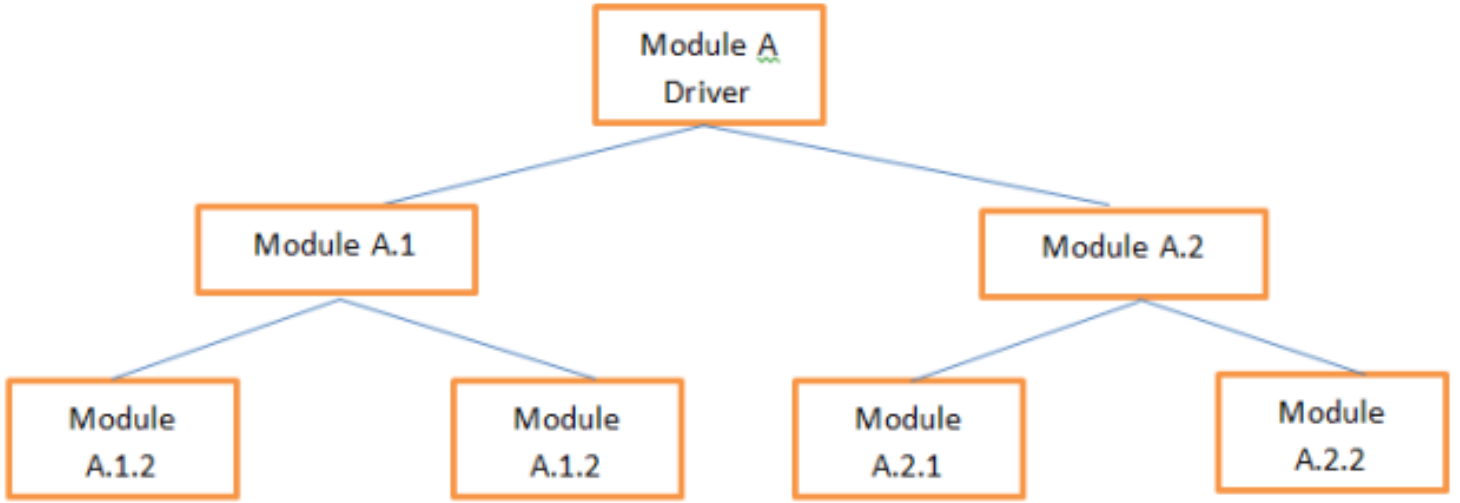
Bottom-up Integration Testing

Bottom up Integration-Example

Step 1- Integrate first set of bottom modules by using Driver A1

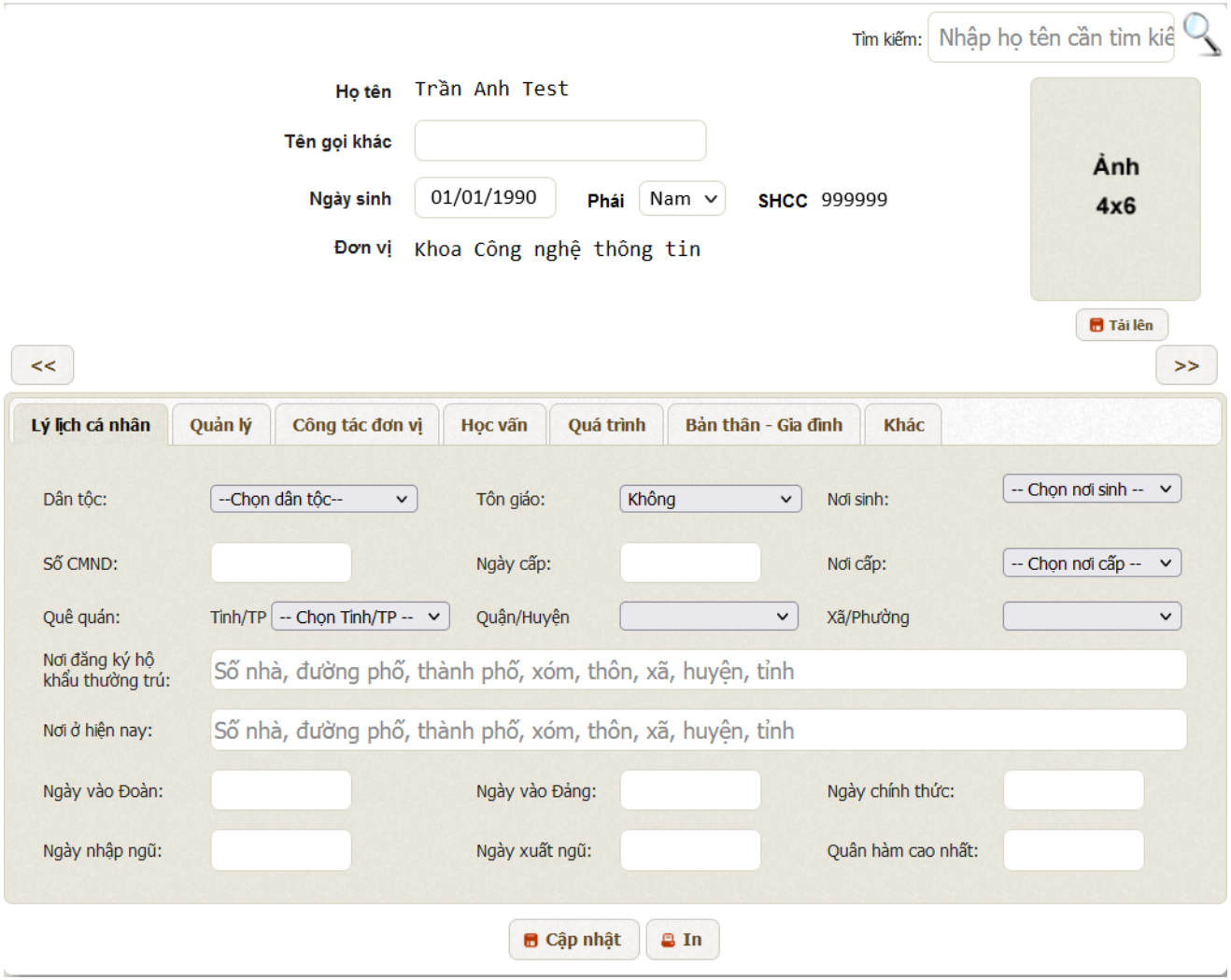
Step 2- Integrate 2nd set of bottom module by using driver A2

Step 3 – Once actual modules for A.1 and A.2 are ready, remove their drivers and integrate entire set of bottom modules by using driver A

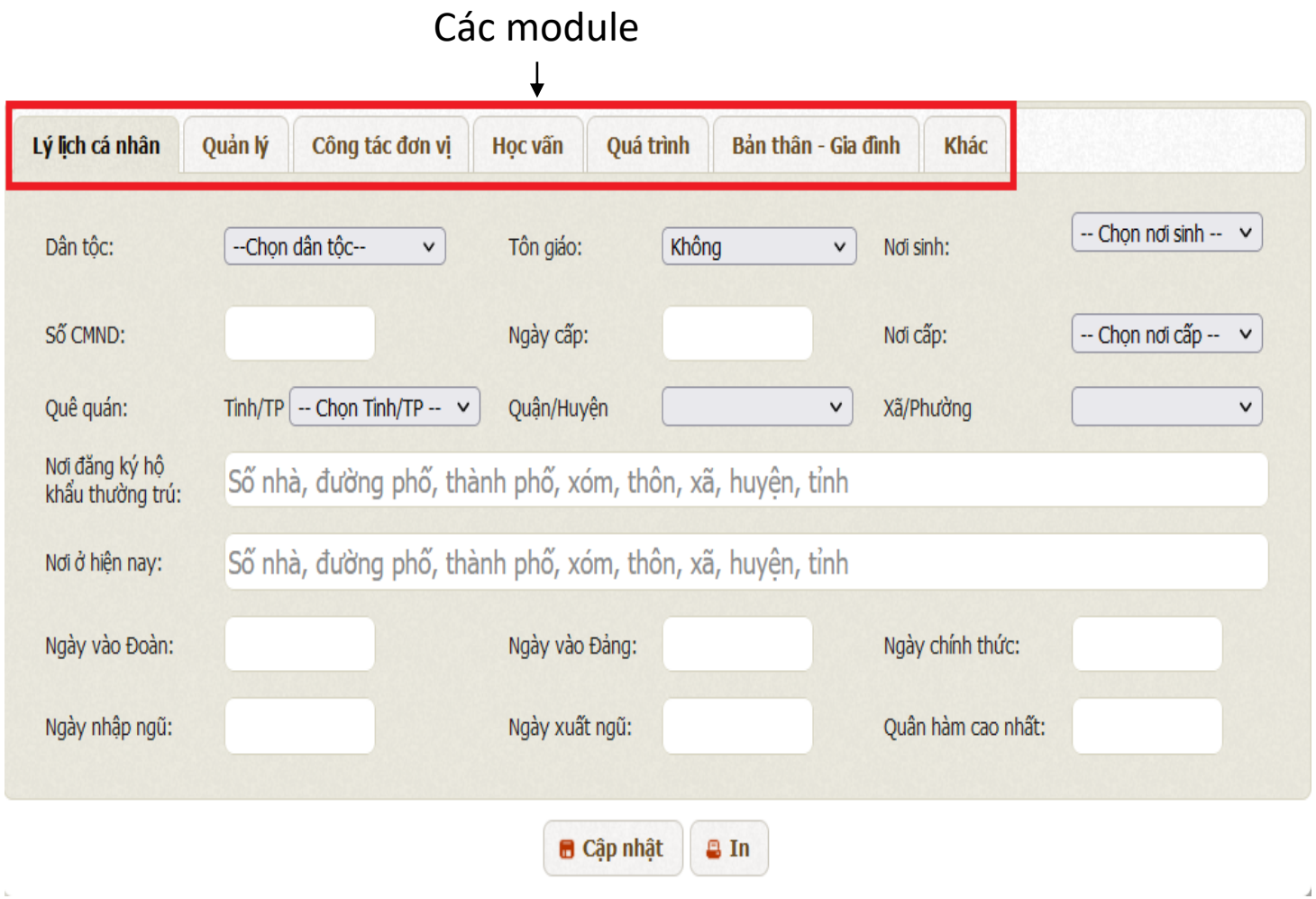


Bottom-up Integration Testing

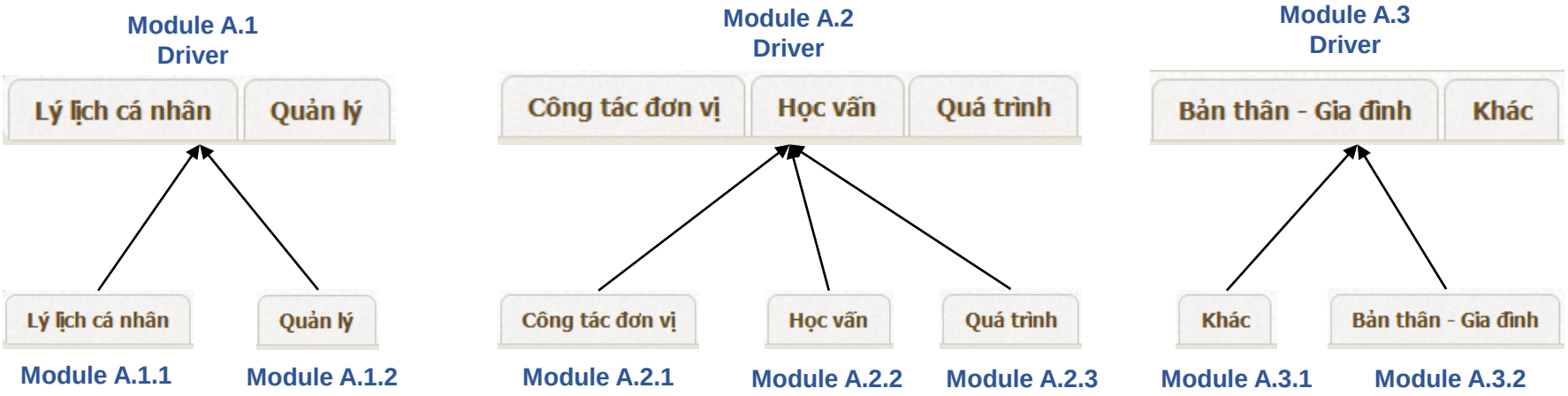
Ví dụ: Chức năng cập nhật thông tin cán bộ



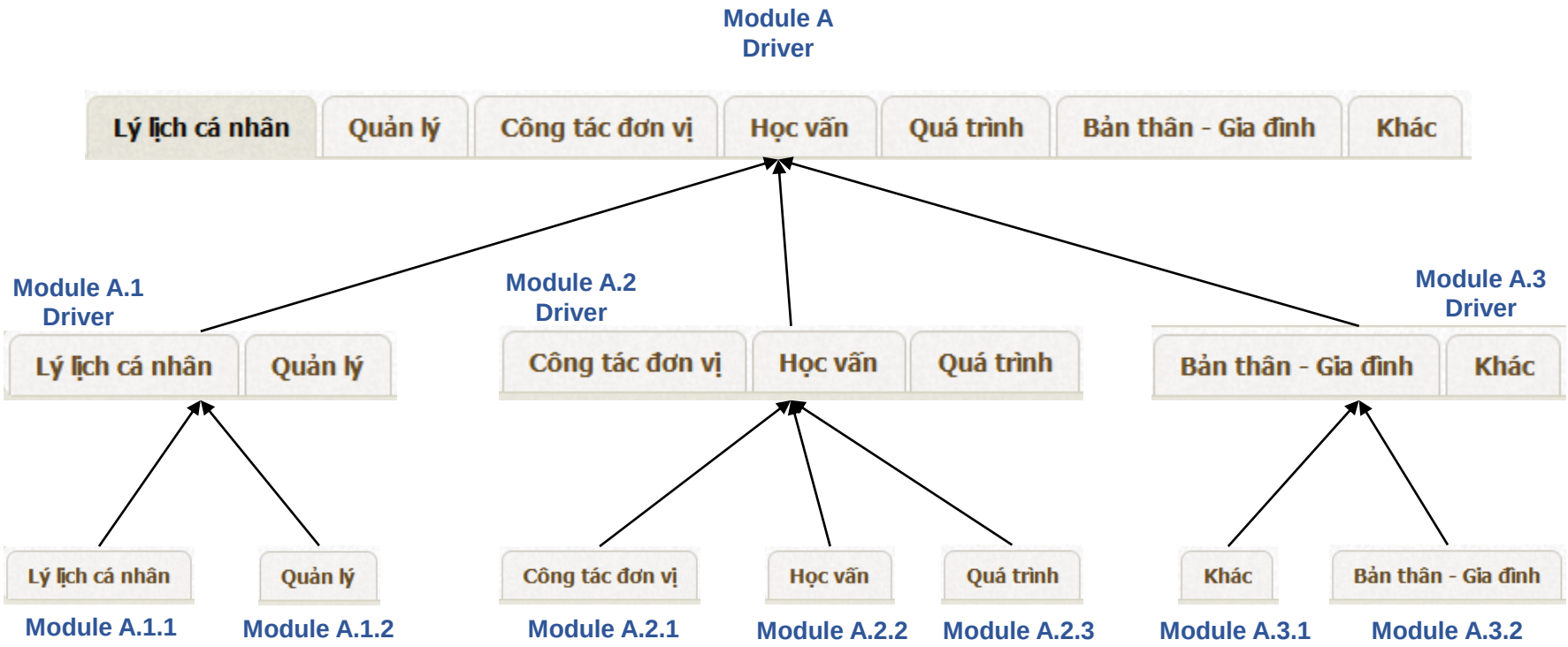
Bottom-up Integration Testing



Bottom-up Integration Testing

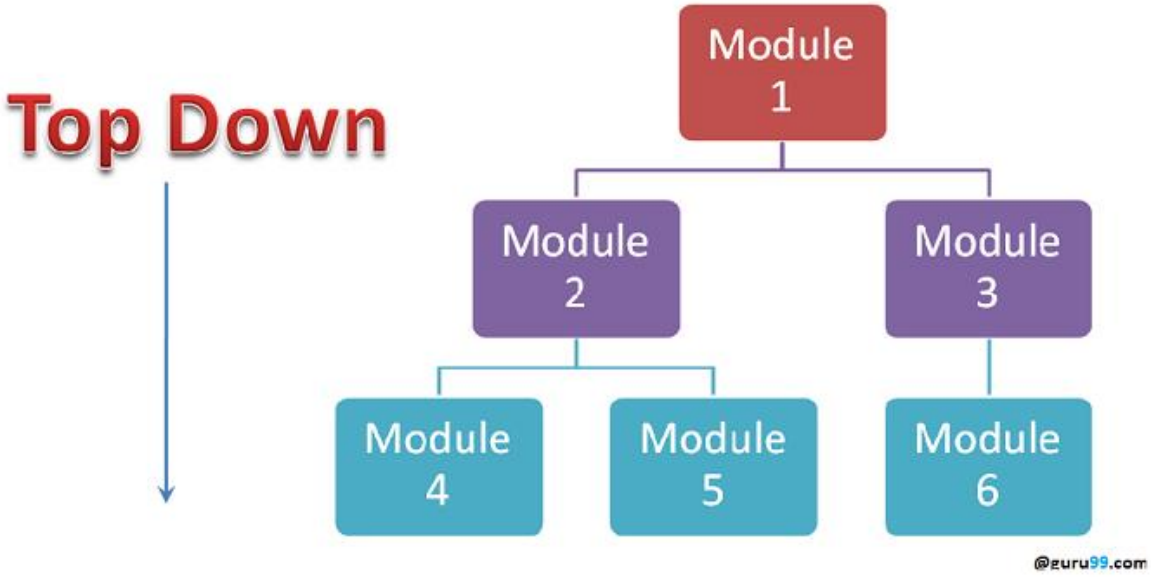


Bottom-up Integration Testing



Top-down Integration Testing

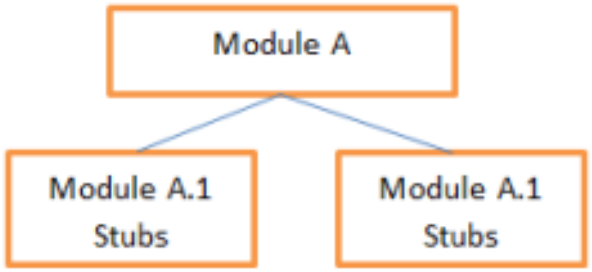
- **Top Down Integration Testing** is a method in which integration testing takes place from top to bottom following the control flow of software system.
- The higher level modules are tested first and then lower level modules are tested and integrated in order to check the software functionality.
- Stubs are used for testing if some modules are not ready.



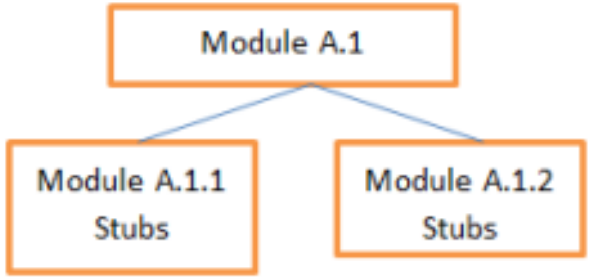
Top-down Integration Testing

Top down Integration - Example

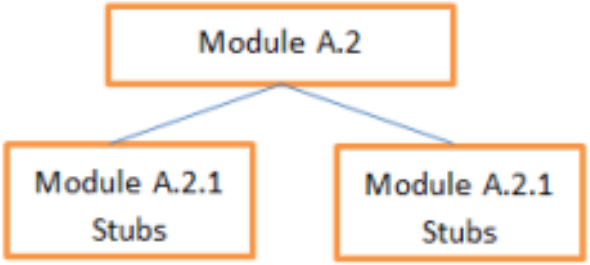
Step1 – Use stubs A.1 and A.2 to test top module A



Step 2 – Replace stubs of A.1 with actual Module and test using stubs for module below



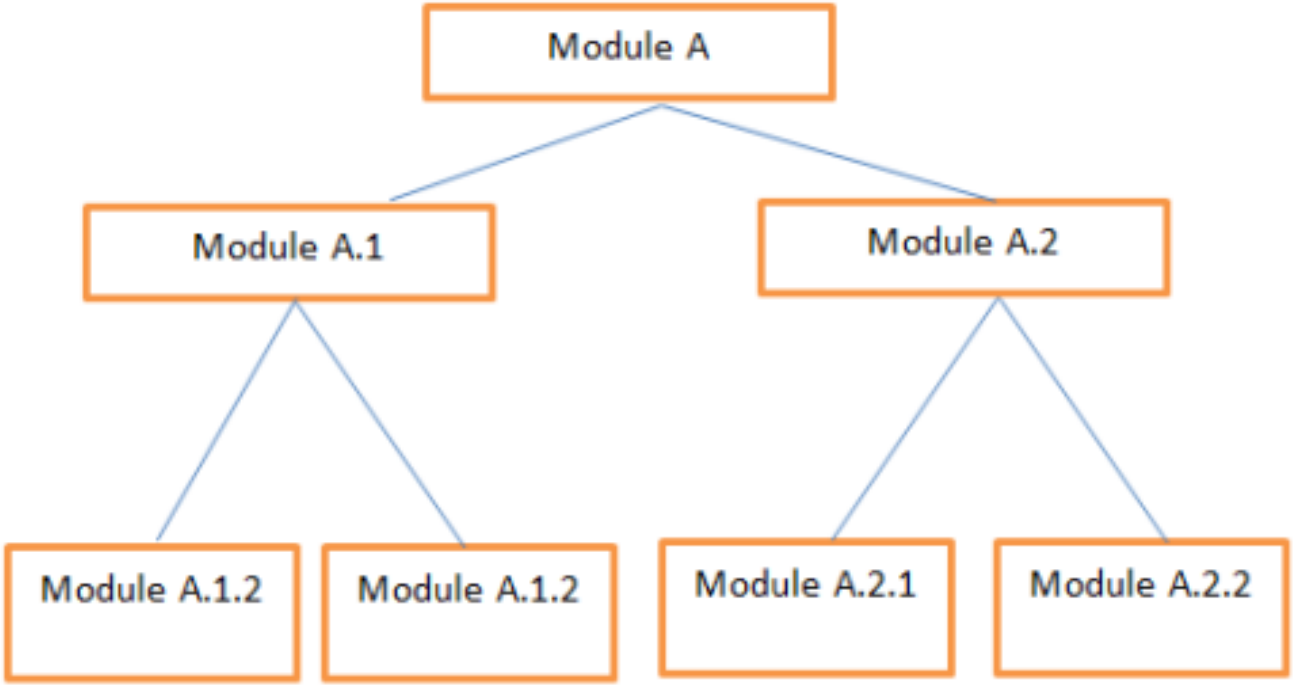
Step 3 - Replace stubs of A.2 with actual Module and test using stubs for module below



Top-down Integration Testing

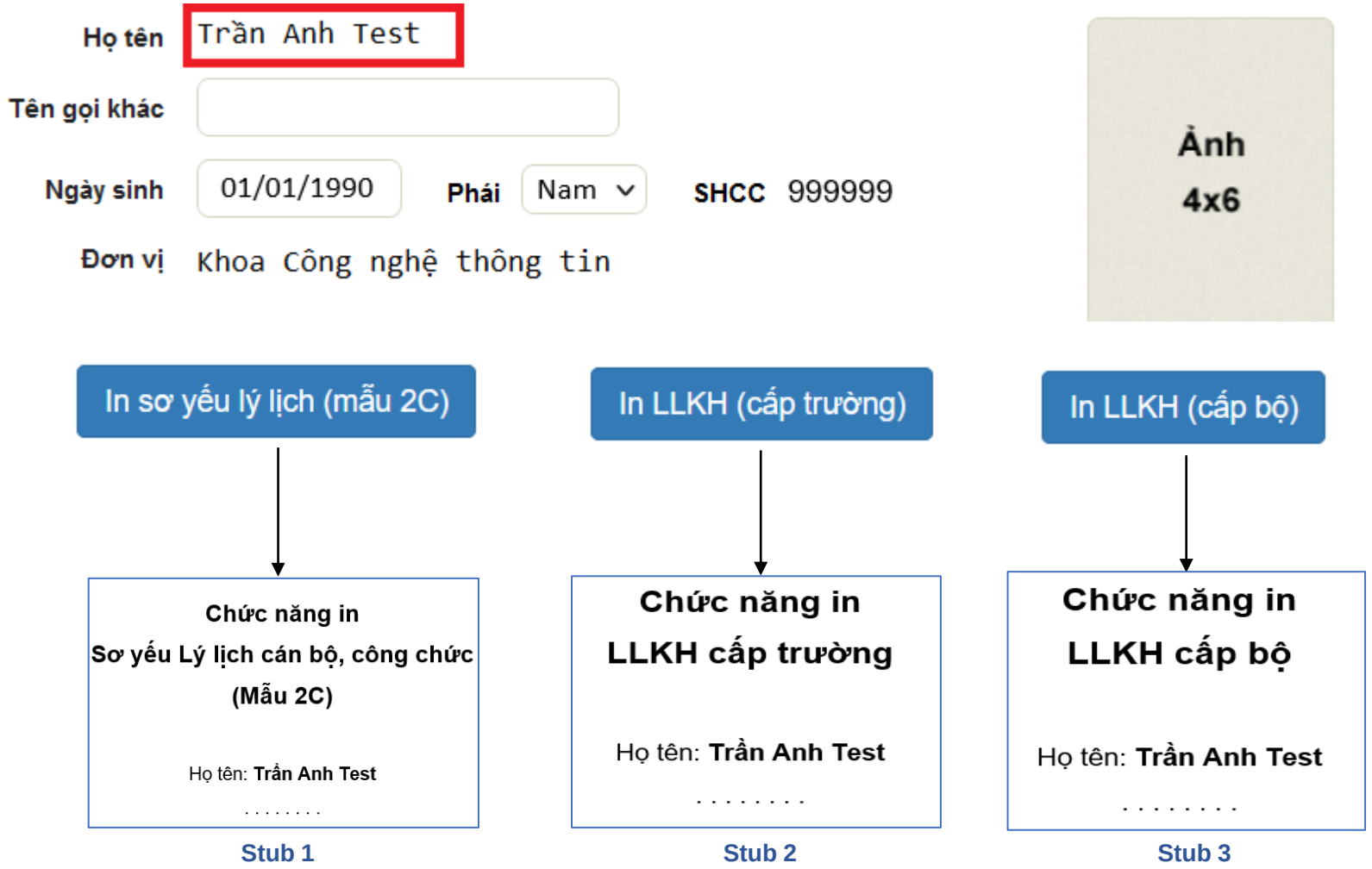
Top down Integration - Example

Step 4 – Replace stubs of bottom most modules with actual modules and test entire hierarchy

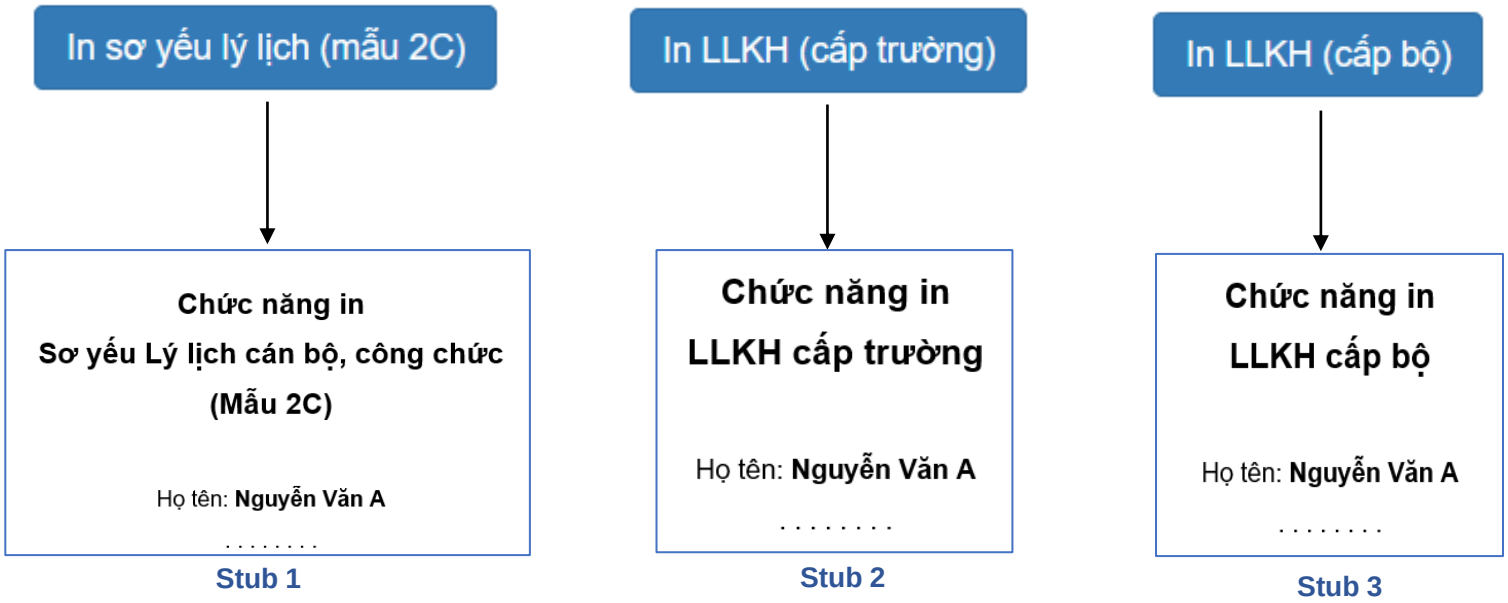


Top-down Integration Testing

Ví dụ:

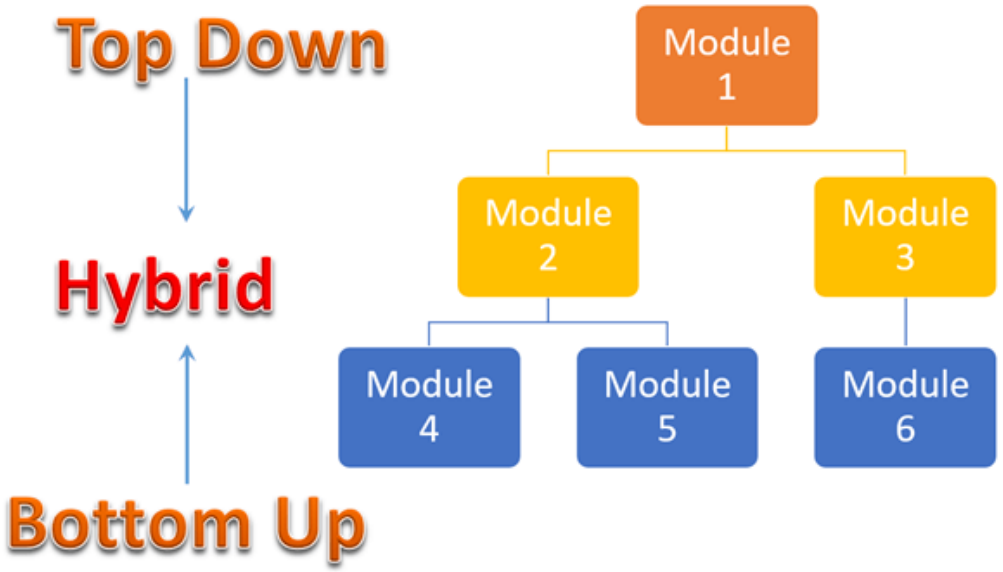


Top-down Integration Testing

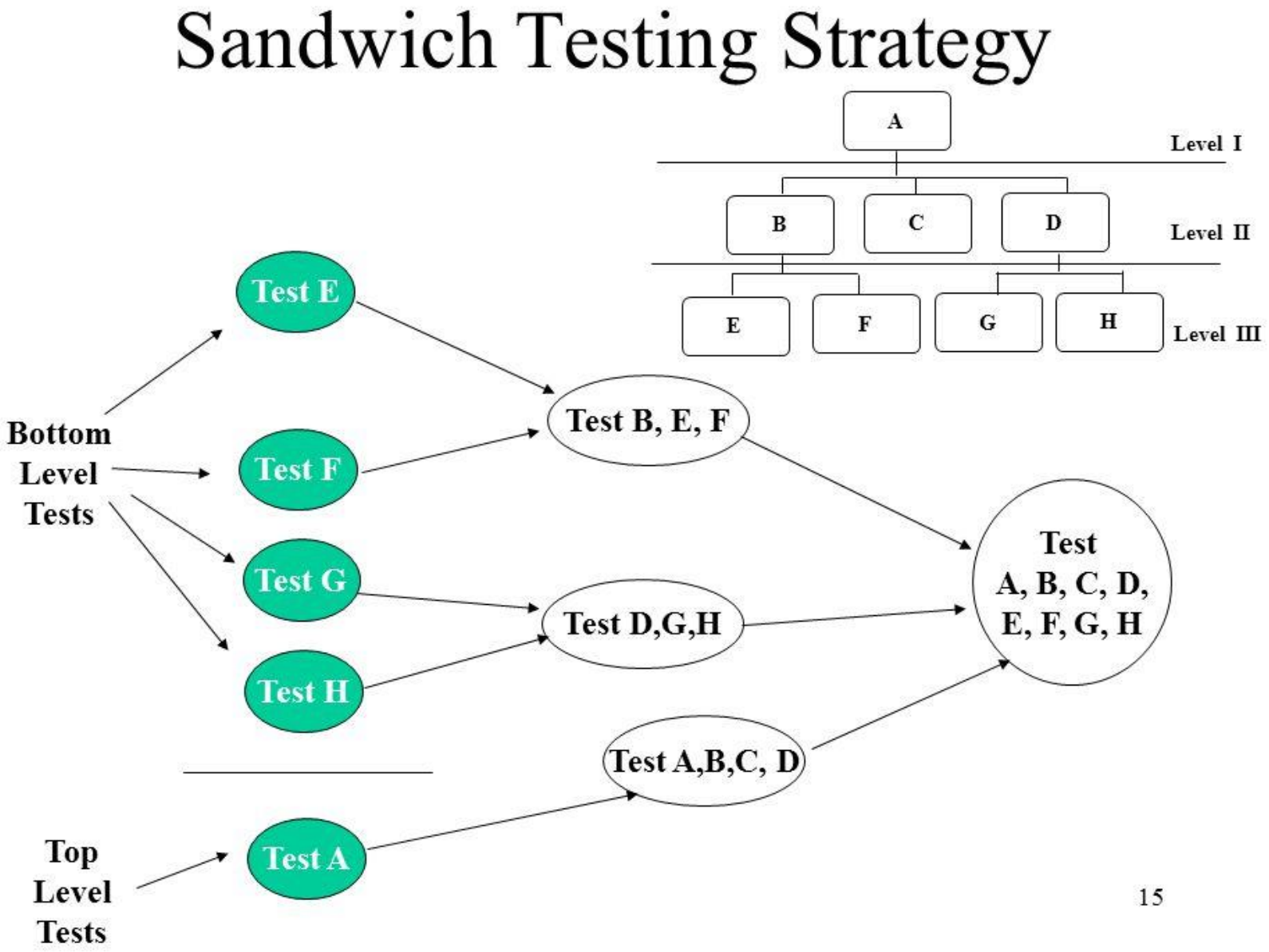


Sandwich Testing

- **Sandwich Testing** is a strategy in which top level modules are tested with lower level modules at the same time lower modules are integrated with top modules and tested as a system.
- It is a combination of Top-down and Bottom-up approaches therefore it is called **Hybrid Integration Testing**.
- It makes use of both stubs as well as drivers.



Sandwich Testing



How to do Integration Testing?

The Integration test procedure irrespective of the Software testing strategies (discussed above):

1. Prepare the Integration Tests Plan
2. Design the Test Scenarios, Cases, and Scripts.
3. Executing the test cases followed by reporting the defects.
4. Tracking & re-testing the defects.
5. Steps 3 and 4 are repeated until the completion of Integration is successful.

Brief Description of Integration Test Plans

- Methods/Approaches to testing
- Scopes and Out of Scopes Items of Integration Testing.
- Roles and Responsibilities.
- Pre-requisites for Integration testing.
- Testing environment.
- Risk and Mitigation Plans.

Best Practices for Integration Testing

- Determine the Integration Test Strategy that could be adopted and later prepare the test cases and test data accordingly.
- Study the Architecture design of the Application and identify the Critical Modules. These need to be tested on priority.
- Obtain the interface designs from the Architectural team and create test cases to verify all of the interfaces in detail. Interface to database/external hardware/software application must be tested in detail.
- After the test cases, it's the test data which plays the critical role.
- Always have the mock data prepared, prior to executing. Do not select test data while executing the test cases.

BÀI TẬP

1. Unit Testing
2. Integration Testing
3. White Box Testing
4. Black-box Testing
5. Driver
6. Stub
7. Incremental Approach
8. Big Bang Approach
9. Top-down Integration
10. Bottom-up Integration

a) Kiểm thử theo theo góc nhìn người sử dụng

b) Kiểm thử theo theo góc nhìn người lập trình

c) Kiểm thử trong đó các module phần mềm được tích hợp một cách hợp lý và được thử nghiệm dưới dạng một nhóm

d) Module có nhiệm vụ kích hoạt các testcase để kiểm thử module đang cần kiểm thử.

e) Một hiện thực ở mức độ tối thiểu nào đó cho 1 module chức năng được dùng bởi module đang cần kiểm thử.

f) Kiểm thử các module chức năng độc lập nhau, sau đó kết hợp chúng lại để tạo ra chương trình.

g) Kiểm tra được thực hiện bằng cách nối hai hoặc nhiều module có liên quan đến logic

h) Mỗi module ở các cấp thấp hơn được kiểm tra với các module cao hơn cho đến khi tất cả các module được kiểm tra

i) Kiểm tra diễn ra từ trên xuống dưới theo luồng điều khiển của hệ thống phần mềm

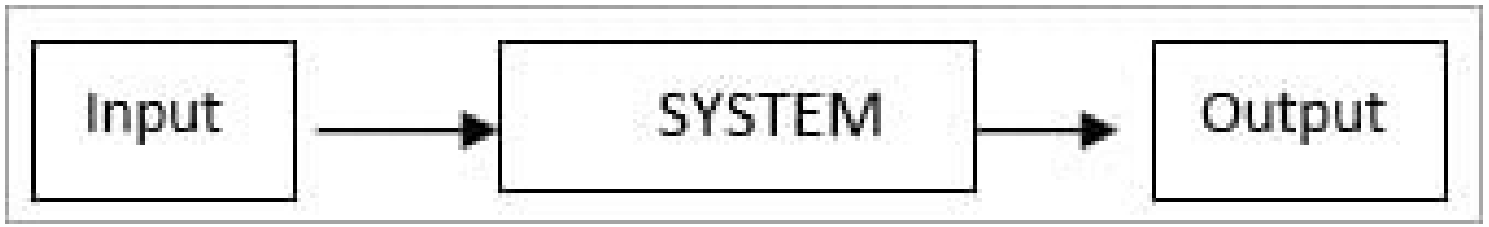
j) Quá trình kiểm thử từng chương trình con, từng thủ tục nhỏ trong chương trình

System Testing

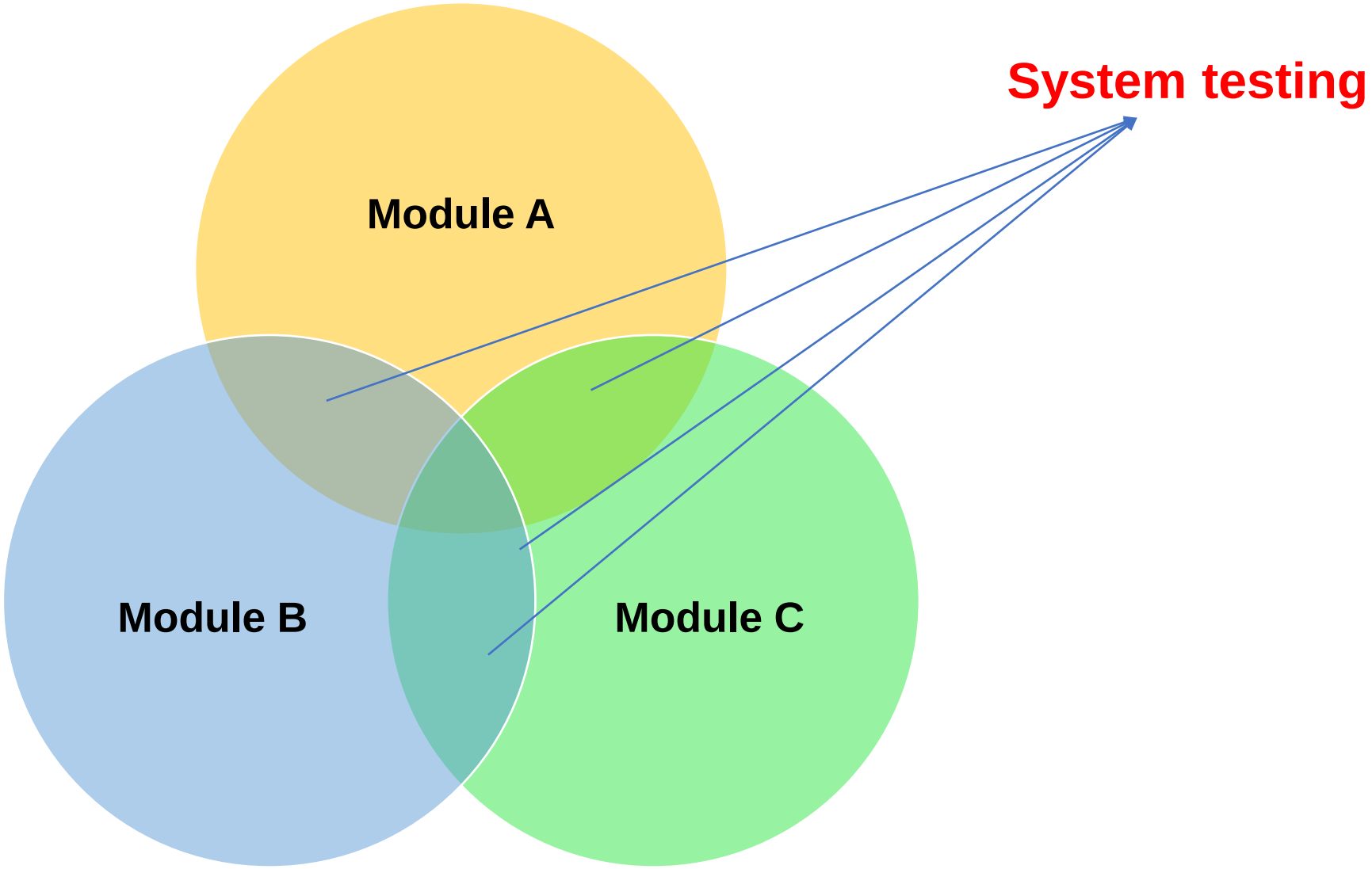
- What is System Testing?
- Approach
- Focus criteria
- How To Perform System Test?
- Advantages
- Entry/Exit Criteria
- System Test Plan
- System Test Cases
- Types Of System Testing

What is System Testing?

- System Testing means testing the system as a whole.
- All the modules/components are integrated in order to verify if the system works as expected or not.
- System Testing is done after Integration Testing.
- This plays an important role in delivering a high-quality product.

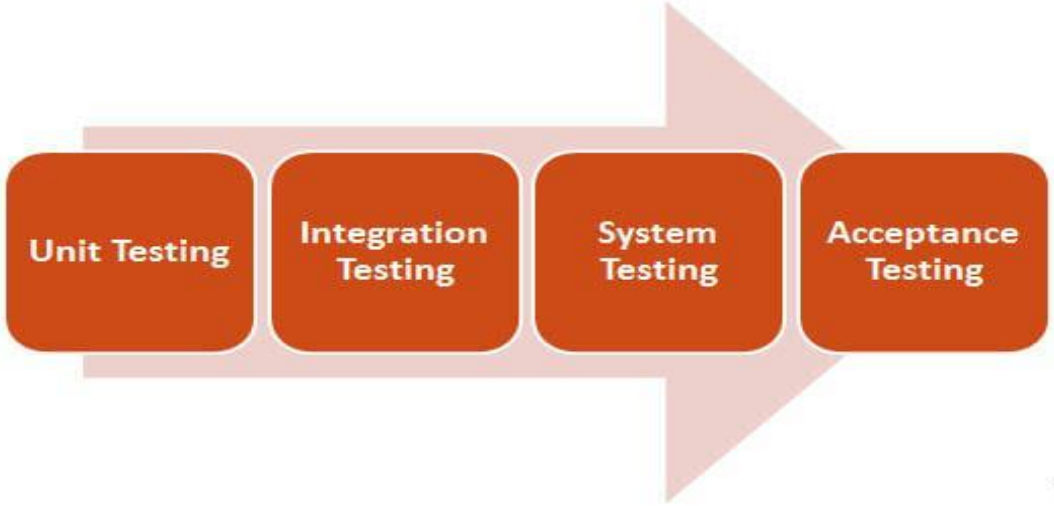


What is System Testing?



Approach

- It is performed when Integration Testing is completed.
- It is mainly a Black-box type testing.
- This testing evaluates the working of the system from a user point of view, with the help of a specification document.
- It does not require any internal knowledge of systems like the design or structure of the code.
- It contains functional and non-functional areas of application/product.



Focus criteria

It mainly focuses on the following:

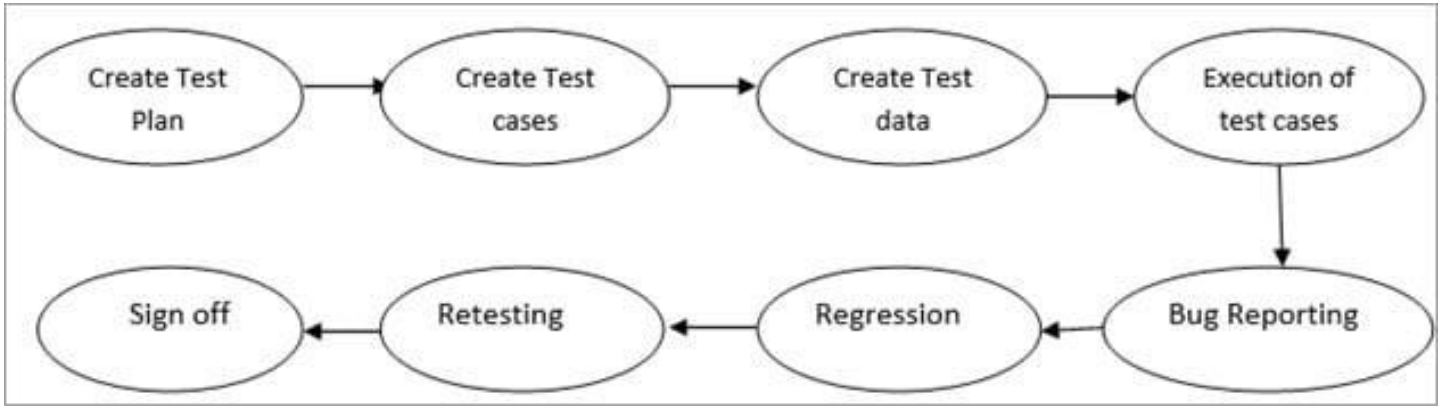
- External interfaces (UI/ UX)
- Multiprogram and complex functionalities
- Security
- Recovery
- Performance
- Operator and user’s smooth interaction with the system
- Installability
- Documentation
- Usability
- Load/Stress

Why is System Testing important?

- What if an online transaction fails after confirmation?
- What if an item placed in a cart of an online site does not allow to place an order?
- What if in a Gmail account creating a new label gives an error on clicking the create tab?
- What if the system crashes and is not able to recover the data as desired?
- What if installing software on the system takes much more time than expected and at the end gives an error?

How To Perform System Test?

- The first step is to create a Test Plan.
- Create System Test Cases and test scripts.
- Prepare the test data required for this testing.
- Execute the system test cases and script.
- Report the bugs. Re-testing the bugs once fixed.
- Regression testing to verify the impact of the change in the code.
- Repetition of the testing cycle until the system is ready to be deployed.
- Sign off from the testing team.



Entry Criteria

- The system should have passed the exit criteria of Integration.
- Test Plan for this testing should be approved & signed off.
- Test cases/scenarios should be ready to be executed.
- Test scripts should be ready to be executed.
- All the non–functional requirements should be available and test cases for the same should have been created.
- The testing environment should be ready.

Exit Criteria

- All the test cases should be executed.
- No critical or Priority or security-related bugs should be in an open state.
- If any medium or low priority bugs are in an open state, then it should be implemented with the acceptance of the customer.
- Exit Report should be submitted.

System Test Plan

- Purpose & Objective is defined for this test.
- Scope
- Test Acceptance Criteria
- Entry/Exit criteria
- Test Schedule
- Test Strategy
- Resources
- Test Environment
- Test Cases
- Assumptions

System Test Cases

System test cases include the below fields in the template:

Test Case ID	Test Suite Name	Description	Steps	Test Data	Expected Result	Actual Result	Pass/Fail	Remarks
TC_01	ABC							
TC_02								

- Test Case ID
- Test Suite name
- Description: Describes the test case to be executed.
- Steps: Step by step procedure to describe how to perform testing.
- Test Data: Dummy data is prepared to test the application.
- Expected Result: Expected result as per the requirement document
- Actual Result: Result after the execution of the test case is provided
- Pass/Fail: Comparison in actual & expected result defines the Pass/fail criteria.
- Remarks

Types Of System Testing

- Kiểm thử hệ thống tập trung vào kiểm thử các yêu cầu phi chức năng. Có 14 yêu cầu phi chức năng sau đây có thể cần kiểm thử (nhưng không phải phần mềm nào cũng đòi hỏi đủ 14 yêu cầu này):
  - Volume Testing
  - Stress Testing
  - Usability Testing
  - Security Testing
  - Performance Testing
  - Storage Testing
  - Configuration Testing
  - Compatibility/Configuration/Conversion Testing
  - Installability Testing
  - Reliability Testing
  - Recovery Testing
  - Serviceability Testing
  - Documentation Testing
  - Procedure Testing

Kiểm thử dung lượng (Volume Test)

- Mục đích của kiểm thử dung lượng là chỉ ra rằng chương trình không thể xử lý khối lượng dữ liệu lớn được đặc tả trong bảng đặc tả mục tiêu chương trình.
- Ví dụ:
  - Chương trình dịch không thể dịch file mã nguồn dài 10MB.
  - Trình liên kết không thể liên kết 1000 module chức năng khác nhau lại.
  - Thử nghiệm tình trạng của trang web âm nhạc khi có hàng triệu người dùng upload bài hát.
  - Trình xem phim không thể chiếu file film dài 15GB.
- Kiểm thử dung lượng thường đòi hỏi rất nhiều tài nguyên, con người lẫn máy tính.

Kiểm thử tình trạng căng thẳng (StressTest)

- Mục đích của kiểm thử tình trạng căng thẳng là chỉ ra rằng chương trình sẽ không thể hoạt động được hay hoạt động không tốt trong tình huống căng thẳng: quá nhiều yêu cầu đồng thời, quá nhiều chương trình khác đang cạnh tranh tài nguyên,...
- Ví dụ:
  - Web server sẽ bế tắc nếu có 100.000 yêu cầu truy xuất trang web đồng thời.
  - HĐH không thể quản lý 1000 process chạy đồng thời.
  - Trình chiếu phim sẽ không chiếu phim mượt và tốt nếu có nhiều chương trình khác cần rất nhiều tài nguyên đang chạy.

Kiểm thử độ khả dụng (UsabililyTest)

- Mục đích của kiểm thử độ khả dụng là chỉ ra các phương tiện/kết quả nhập/xuất không phù hợp, thân thiện với người dùng:
  - Mỗi đối tượng giao diện có thân thiện, tự nhiên và dễ dùng không?
  - Kết quả hiển thị có ngắn gọn, trong sáng, nghĩa dễ hiểu không?
  - Các cảnh báo có dễ hiểu không ? “IEK022A OPEN ERROR ON FILE ‘SYSIN’ ABEND CODE=102?”
  - Nói chung tất cả các kết quả, các cảnh báo đều phải nhất quán, đồng nhất về cú pháp, về định dạng, ngay cả các từ viết tắt được dùng.

Kiểm thử độ khả dụng (UsabililyTest)

- Một số chú ý :
  - Khi độ chính xác là rất quan trọng như trong hệ thống quản lý ngân hàng, thì thông tin nhập có tính dư thừa đủ không?
  - Hệ thống có quá nhiều nhiệm ý hay các nhiệm ý được người dùng thích dùng không?
  - Hệ thống có trả về đủ đáp ứng với mọi hoạt động nhập?
  - Chương trình có dễ dùng và thân thiện?

Kiểm thử các dịch vụ cộng thêm (Serviceability Test)

- Trong mục tiêu của phần mềm có thể đề cập đến 1 số dịch vụ cộng thêm, ví dụ như :
  - Chương trình hỗ trợ xuất báo cáo dạng file excel.
  - Thời gian trung bình để khắc phục lỗi khi có sự cố xảy ra.
  - Các thủ tục bảo trì.
  - Chất lượng của tài liệu luận lý bên trong.
- Các mục tiêu trên, nếu có đề cập trong mục tiêu chương trình thì cần phải được kiểm thử.

Kiểm thử tính an ninh (Security Test)

- An ninh phần mềm gồm 3 vấn đề chính là bảo mật, tính toàn vẹn dữ liệu và độ sẵn sàng đáp ứng.
- Nghiên cứu các vấn đề liên quan đến an ninh trong các hệ thống tương tự rồi tạo các testcase để chứng minh rằng các vấn đề này cũng tồn tại trong chương trình cần kiểm thử.
- Các ứng dụng mạng và ứng dụng theo công nghệ Web hiện nay cần được kiểm thử tính an ninh ở mức độ cao hơn nhiều so với phần mềm truyền thống trên máy đơn. Điều này đặc biệt đúng cho các website thương mại, ngân hàng...

Kiểm thử hiệu suất làm việc (Performance Test)

- Mục đích của kiểm thử hiệu suất làm việc là chỉ ra rằng phần mềm không đạt được hiệu xuất được đặc tả trong mục tiêu chương trình.
- Ví dụ :
  - Trình chiếu phim full HD không chiếu kịp 20 frame/sec.
  - Trình nén dữ liệu không nén dữ liệu kịp với tốc độ đề ra.
  - Trình soạn thảo văn bản không nhận và xử lý kịp các ký tự được nhập bởi người dùng.
  - Trình ghi DVD không tạo dữ liệu ghi kịp tốc độ mà ổ DVD yêu cầu...

Kiểm thử độ sử dụng bộ nhớ (Storage Test)

- Mục đích của kiểm thử độ sử dụng bộ nhớ là chỉ ra rằng phần mềm không tuân thủ về dung lượng bộ nhớ tối thiểu/tối đa được đặc tả trong mục tiêu chương trình.
- Ví dụ:
  - Kích thước tối thiểu 128KB không đủ để chạy chương trình.
  - Chương trình không dùng hết kích thước bộ nhớ tối đa là 4GB.
  - Chương trình không chạy được khi đĩa còn dung lượng trống tối thiểu là 4MB.
  - Chương trình không quản lý được dung lượng đĩa là 1TB...

Kiểm thử cấu hình làm việc (Configuration Test)

- Nhiều chương trình như HĐH, hệ quản trị CSDL, Website,...thường sẽ làm việc được trên nhiều cấu hình phần cứng/phần mềm cấp thấp.
- Số lượng các cấu hình khác nhau có thể quá lớn, nhưng ta nên chọn 1 số cấu hình phổ dụng nhất để kiểm thử xem chương trình có chạy tốt trên các cầu hình này không.

Kiểm thử tính tương thích/chuyển đổi/cấu hình

(Compatibility/Configuration/Conversion Test)

- Đời sống của 1 phần mềm thường dài, nhất là phần mềm thương mại của các hãng lớn.
- Trong cuộc đời của mình, phần mềm được phát triển tăng dần theo từng release, từng version.
- Về nguyên tắc, version mới sẽ tương thích ngược với version đã có.
- Mức độ tương thích, khả năng chuyển đổi định dạng file dữ liệu từ cũ sang mới hay ngược lại, khả năng cấu hình version mới để có thể làm việc như version cũ,... có thể được đặc tả trong mục tiêu của chương trình.
- Nếu có thì ta phải kiểm thử các đặc tả này xem version cần kiểm thử có đáp ứng được không.
- Ví dụ : Word 2003 có thể cấu hình để chạy y như Word 97 không ?

Kiểm thử khả năng cài đặt (Installability Test)

- Một số hệ thống phần mềm có thủ tục cài đặt khá phức tạp.
- Chương trình cài đặt chạy sai có thể ngăn chặn người dùng không dùng được phần mềm được cài đặt.
- Nhiệm vụ của kiểm thử khả năng cài đặt là kiểm thử chương trình cài đặt có hoạt động đúng không?

Kiểm thử độ tin cậy (Reliability Test)

- Mục tiêu của mọi loại kiểm thử đều hướng đến việc cải tiến độ tin cậy của chương trình.
- Nếu mục tiêu của chương trình chứa các yêu cầu đặc biệt về độ tin cậy, ta cũng cần phải thực hiện hoạt động kiểm thử độ tin cậy đặc thù.
- Việc kiểm thử các mục tiêu về độ tin cậy có thể khó khăn.
- Ví dụ: 1 hệ thống online hiện đại như WAN hay ISP thường có thời gian làm việc thực tế bằng 99.97% thời gian sống của nó.
- Chưa có cách để có thể kiểm thử mục tiêu này với thời gian kiểm thử hàng tháng hay hàng năm.

Kiểm thử độ phục hồi sau lỗi (Recovery Test)

- Các chương trình như hệ điều hành, hệ quản trị database, các chương trình xử lý từ xa thường có các mục tiêu về phục hồi sau lỗi để miêu tả cách hệ thống phục hồi sau khi lỗi dữ liệu, lỗi phần mềm hay lỗi phần cứng xảy ra.
- Mục tiêu của kiểm thử độ phục hồi sau lỗi:
  - Chỉ ra rằng các chức năng phục hồi không làm việc đúng.
  - Chỉ ra rằng hệ thống không thỏa sự thỏa thuận về thời gian
  - Trung bình để phục hồi sau lỗi (MTTR).

Kiểm thử tài liệu (Documentation Test)

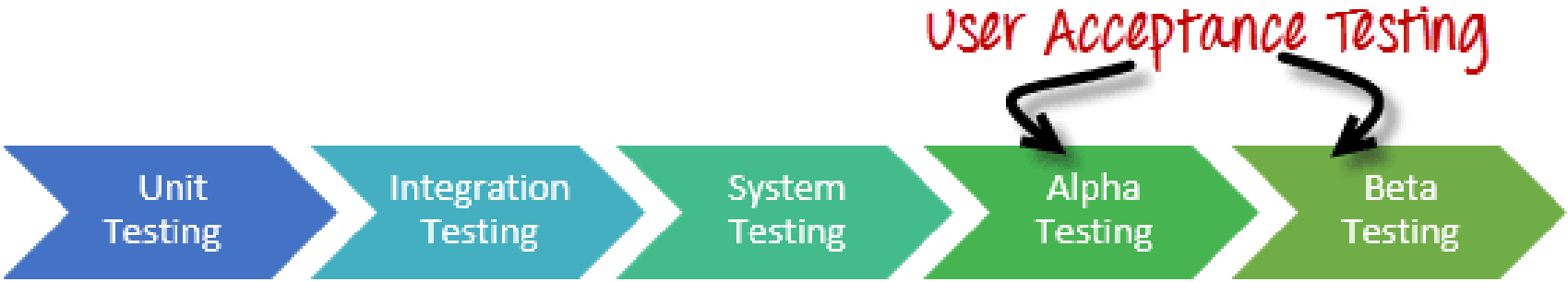
- Kiểm thử hệ thống cũng có liên quan đến độ chính xác của tài liệu dành cho người dùng.
- Cách chính yếu để thực hiện điều này là dùng tài liệu để xác định các testcase hệ thống có độ ưu tiên cao.
- Tài liệu dành cho người dùng nên là chủ đề của 1 hoạt động thanh tra (tương tự như khái niệm thanh tra mã nguồn), hãy kiểm tra nó để biết được độ chính xác và tính trong sáng

Kiểm thử thủ tục (Procedure Test)

- Nhiều phần mềm là thành phần của hệ thống lớn hơn nhưng chưa được tự động hóa hoàn toàn liên quan đến nhiều thủ tục mà con người cần thực hiện.
- Bất kỳ thủ tục của con người được kê ra, như thủ tục dành cho người quản trị hệ thống, quản trị database, người dùng đầu cuối nên được kiểm thử trong suốt hoạt động kiểm thử hệ thống.

Acceptance Testing

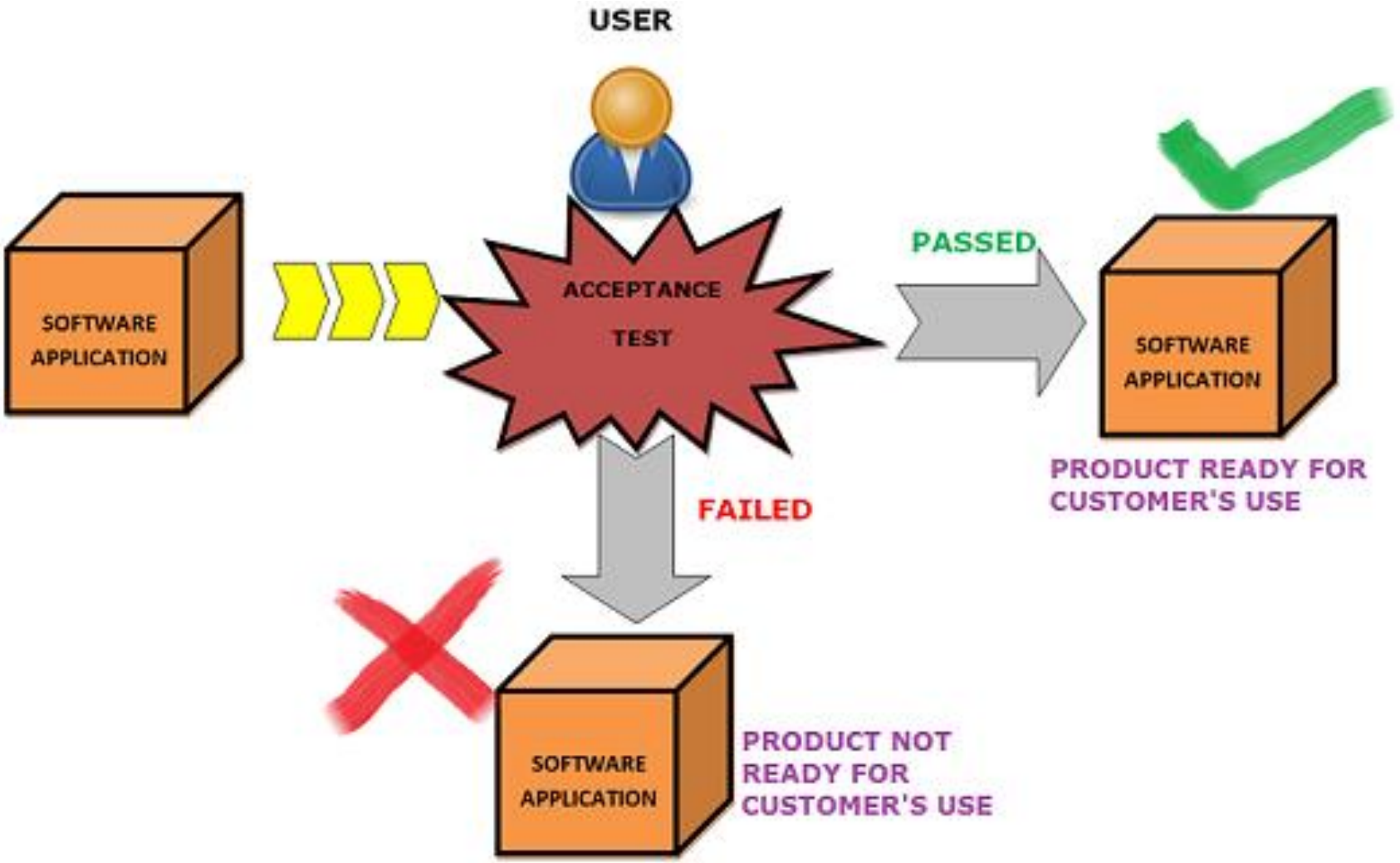
- What is Acceptance Testing?
- What is User Acceptance Testing?
- Alpha Testing
- Beta Testing



What is Acceptance Testing?

- It is a formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to check if the system is acceptable to the users.
- Acceptance testing a pure functional testing to check the system behavior using the real data. It is also called *business user testing.*

What is Acceptance Testing?

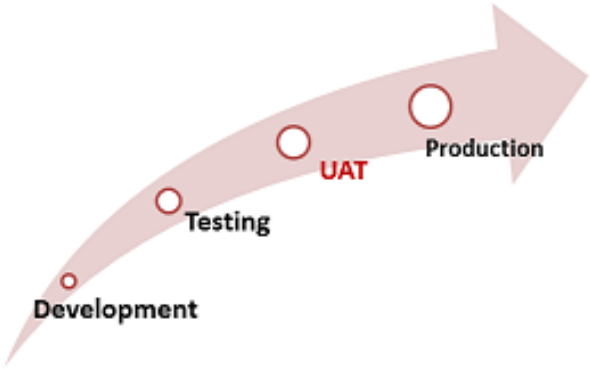


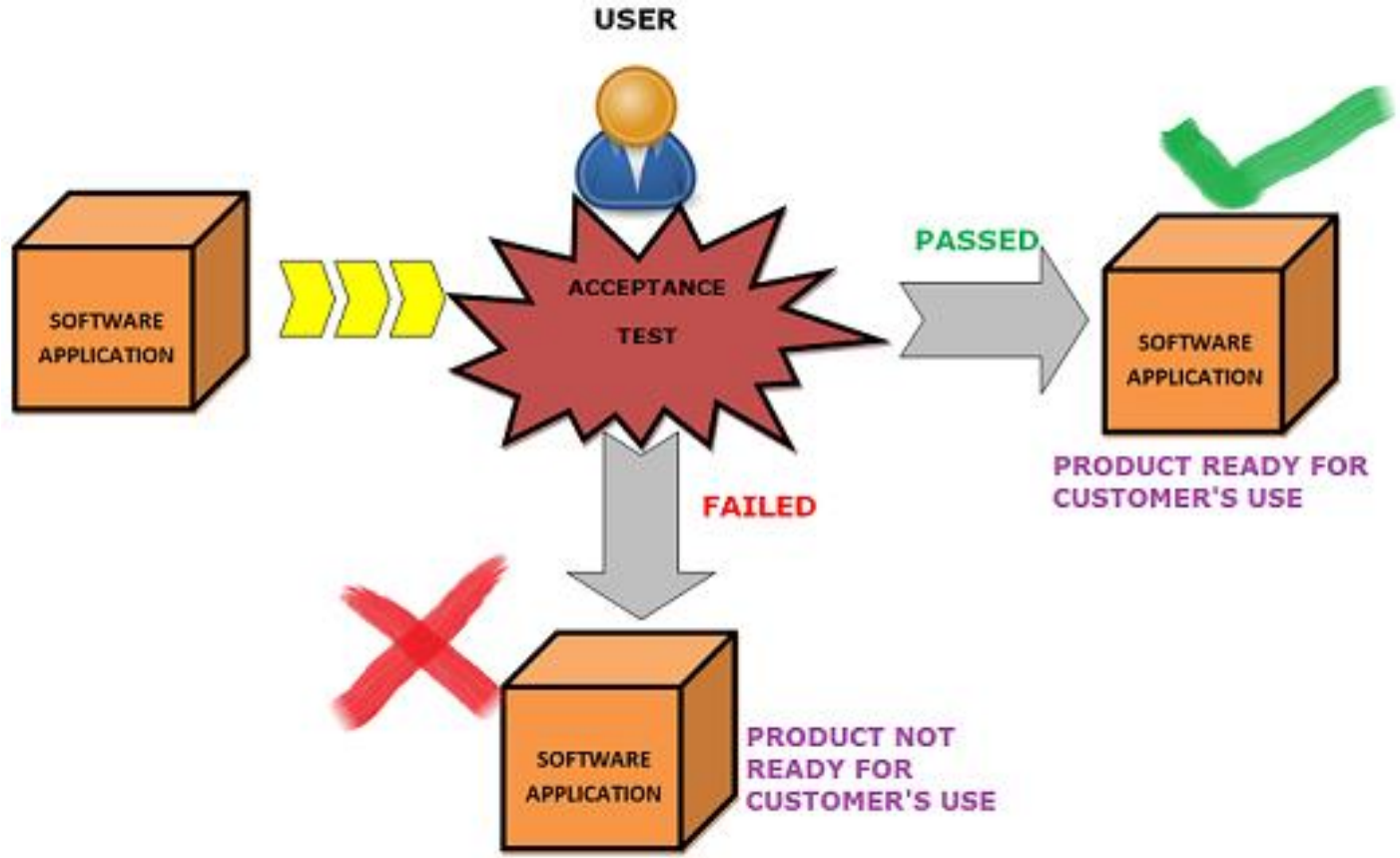
What is User Acceptance Testing?

- **User Acceptance Testing (UAT)** is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment.
- UAT is done in the final phase of testing after functional, integration and system testing is done.
- The main purpose of UAT is to validate end to end business flow.
- User Acceptance Testing is carried out in a separate testing environment with production-like data setup.

Who Performs UAT?

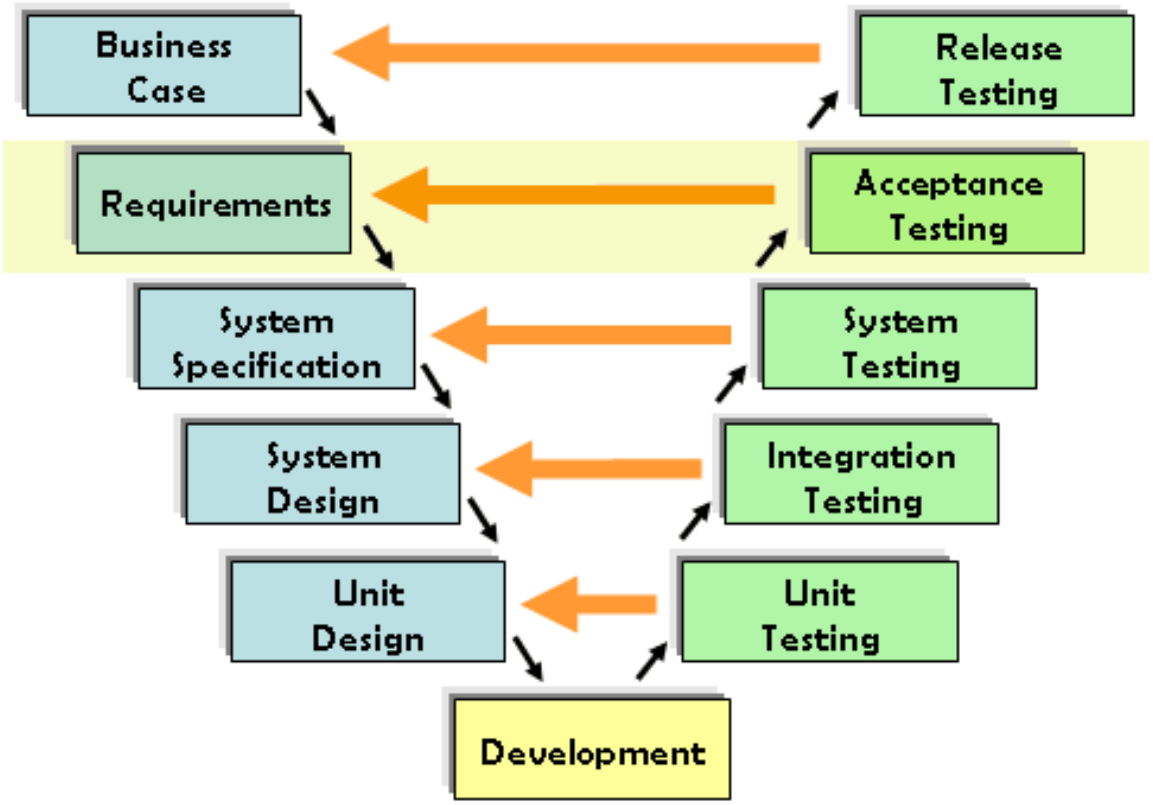
- Client
- End users





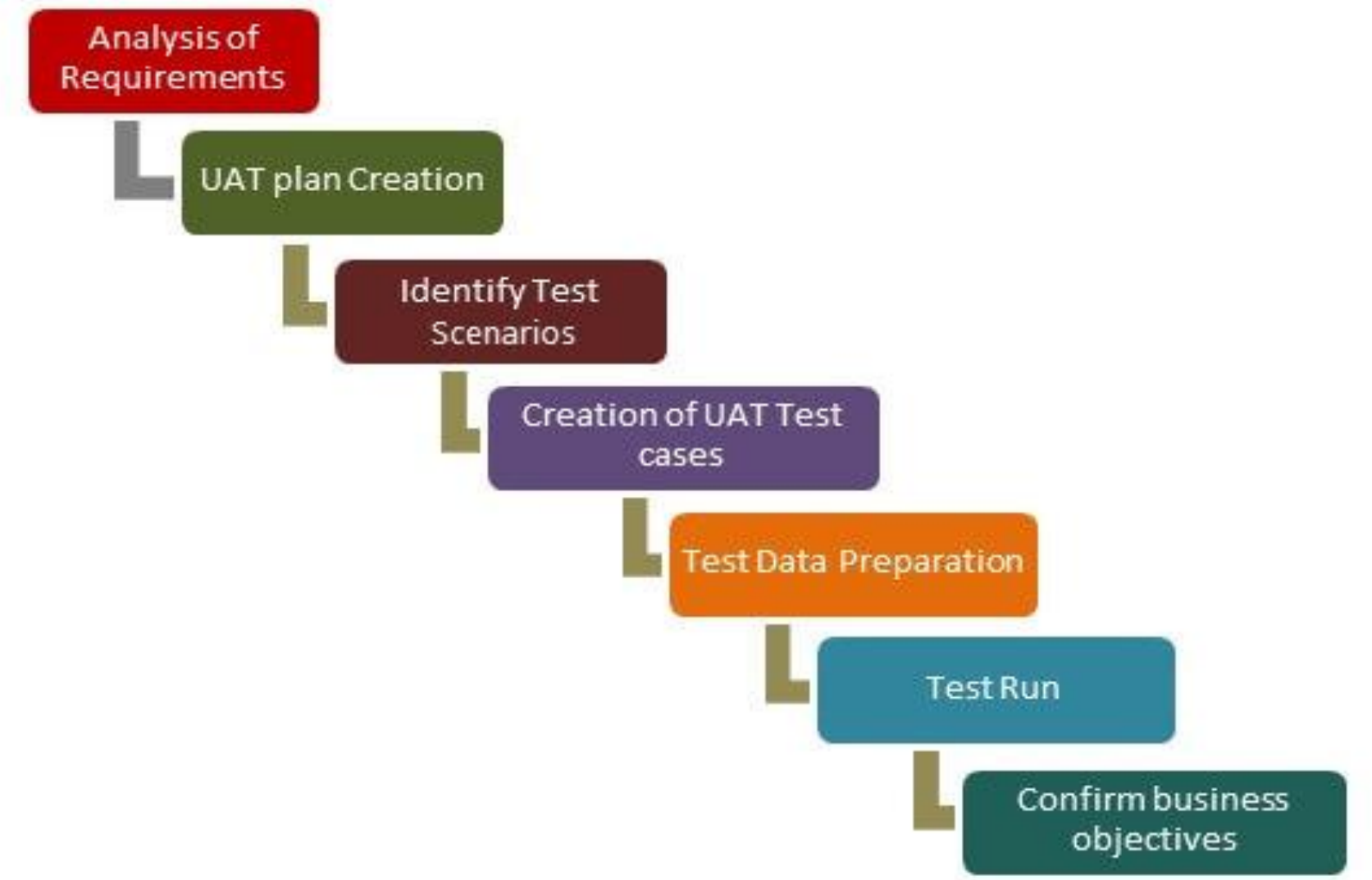
Acceptance Testing and V-Model

In V-Model, User acceptance testing corresponds to the requirement phase of the Software Development life cycle (SDLC).



How to do UAT Testing?

UAT is done by the intended users of the system or software.



1. Analysis of Business Requirements

One of the most important activities in the UAT is to identify and develop test scenarios. These test scenarios are derived from the following documents

- Project Charter
- Business Use Cases
- Process Flow Diagrams
- Business Requirements Document(BRD)
- System Requirements Specification(SRS)

2. Creation of UAT Plan

- The UAT test plan outlines the strategy that will be used to verify and ensure an application meets its business requirements.
- It documents entry and exit criteria for UAT, Test scenarios and test cases approach and timelines of testing.

3. Identify Test Scenarios and Test Cases

- Identify the test scenarios with respect to high-level business process and create test cases with clear test steps.
- Test Cases should sufficiently cover most of the UAT scenarios.
- Business Use cases are input for creating the test cases.

4. Preparation of Test Data

- It is best advised to use live data for UAT.
- Data should be scrambled for privacy and security reasons.
- Tester should be familiar with the database flow.

5. Run and record the results

- Execute test cases and report bugs if any. Re-test bugs once fixed.
- Test Management tools can be used for execution
  - JIRA
  - Klaros
  - qTest

6. Confirm Business Objectives met

- Business Analysts or UAT Testers needs to send a sign off mail after the UAT testing.
- After sign-off, the product is good to go for production.
- Deliverables for UAT testing are Test Plan, UAT Scenarios and Test Cases, Test Results and Defect Log.

Best Practices

- Prepare UAT plan early in the project life cycle
- Prepare Checklist before the UAT starts
- Conduct Pre-UAT session during System Testing phase itself
- Set the expectation and define the scope of UAT clearly
- Test End to End business flow and avoid system tests
- Test the system or application with real-world scenarios and data
- Think as an Unknown user to the system
- Perform Usability Testing
- Conduct Feedback session and meeting before moving to production.

Alpha testing

- This is a form of internal acceptance testing performed mainly by the in-house software QA and testing teams.
- Alpha testing is the last testing done by the test teams at the development site after the acceptance testing and before releasing the software for the beta test.
- Alpha testing can also be done by the potential users or customers of the application. But still, this is a form of in-house acceptance testing.

Beta testing

- This is a testing stage followed by the internal full alpha test cycle.
- This is the final testing phase where the companies release the software to a few external user groups outside the company test teams or employees.
- This initial software version is known as the beta version.
- Most companies gather user feedback in this release.
- **In short, beta testing can be defined as** – the testing carried out by real users in a real environment.

TỔNG KẾT CHƯƠNG 3

- **Software Testing Levels**
- **Unit Testing**
- **Integration Testing**
- **System Testing**
- **Acceptance Testing**

Chương kế tiếp...

CÁC PHƯƠNG PHÁP KIỂM THỬ PHẦN MỀM