Innovative Entrepreneurship

Lecture # 11

Saltanat Kondybayeva, PhD

Topic 11.
Examination of innovative projects

P L A N

1. Necessary of Examination
2. Types of Testing (Examination)

Necessary of Examination

Examination allows us to reveal the most effective innovative projects, and to estimate the results of their implementation.

An innovative project should finally provide scientific and technical progress, as well as the economic growth that will positively affect the quality of human life.

Examination of innovative projects is limited to evaluation of the technological level of new products and the technological processes of their manufacture. The criteria for this examination include: novelty, competitiveness, economic efficiency, terms of implementation of the innovation, etc.

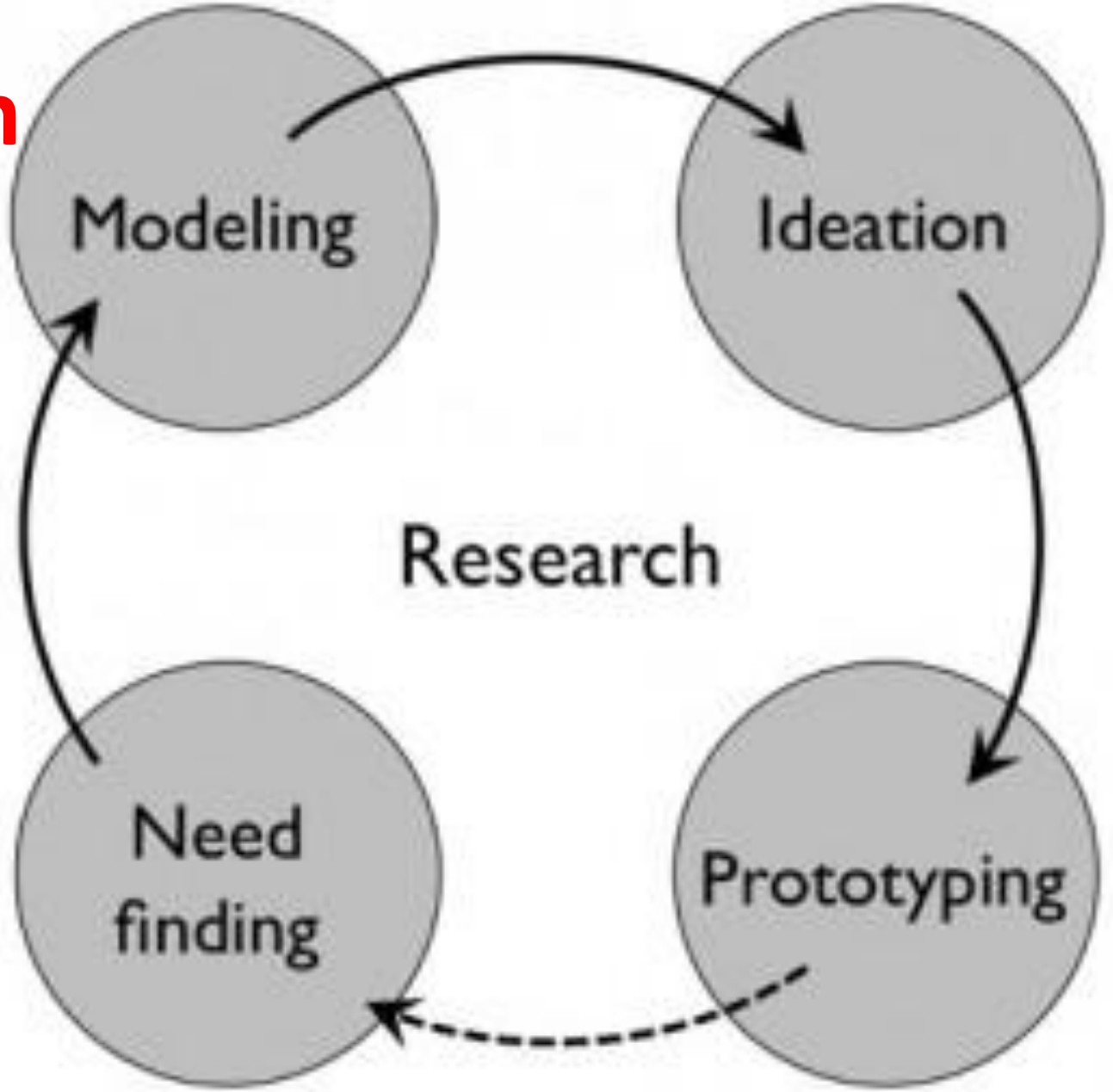
The goal of the efficiency examination of an innovative project is not only the evaluation of its scientific and technological levels, but the resources and capability of its implementation as well.

Therefore, the examination should be performed through quality to understand whether the organization which has conceived the innovation, or plans its application, is capable of implementing the project. In fact, the final result of innovative activity depends on both scientific and technical levels, and the conditions in which it is carried out.

Experimentation

- The experimentation stage tests an idea, such as with a prototype or pilot test.
- Experimentation can remain continuous or exist in spurts, as advocates and screeners reevaluate an idea. Sometimes, experimentation leads to new ideas due to information that is gathered on the results and the overall feasibility of the original idea.
- Time is crucial in this process; individuals must be given adequate time to run the experiments. As refinements and evaluations occur, they must be given enough time to reflect on the experiments.

Experimentation



Good ideas - Bad Products

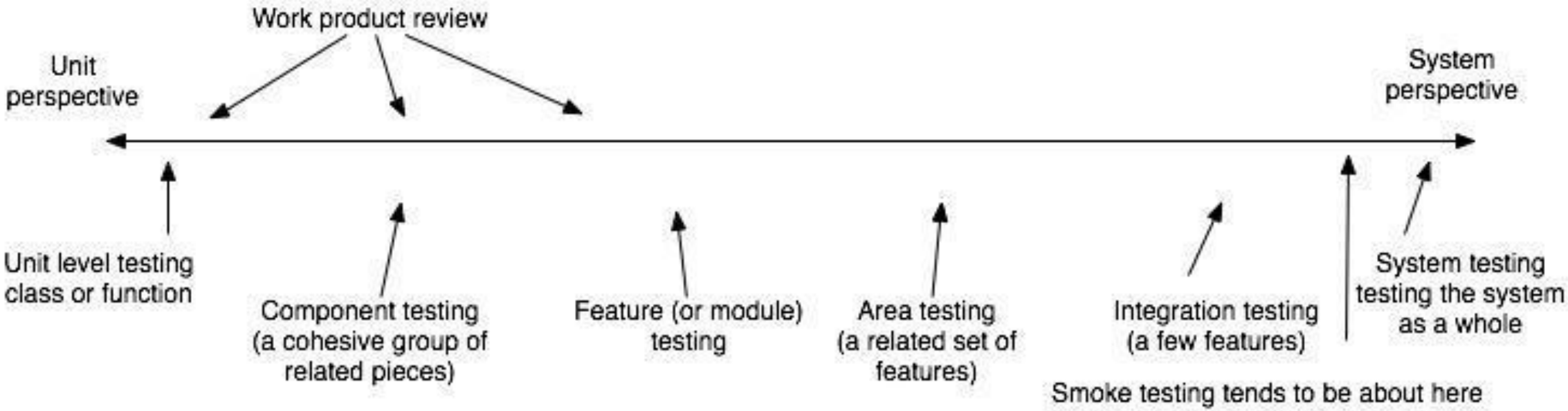
- Not viable manufacturing techniques
- Expensive
- Apprehensive market
- Ahead of its time

up-to-date quality management techniques

- quality management systems;
- environment management systems;
- self-assessment;
- assessment results for quality contests.

What You Need to Know About Testing

Project managers need to realize that only people can do everything above the line in the testing continuum (see figure 1 below). People read work products, using any of the reviewing techniques: inspections, reviews, walkthroughs, buddy reviews, or pairing. Since only people can perform this work, project teams tend to postpone work product review, unless it's built into how people do their work (like pairing), or if the time to do this work is built into the project schedule. The more serial the lifecycle, the more project managers need to help the team make time to do work product review, because it's the earliest feedback about the project's progress anyone can get.


Work product review
Unit perspective
System perspective
Unit level testing class or function
Component testing (a cohesive group of related pieces)
Feature (or module) testing
Area testing (a related set of features)
Integration testing (a few features)
System testing testing the system as a whole
Smoke testing tends to be about here

Types of Testing (Examination)

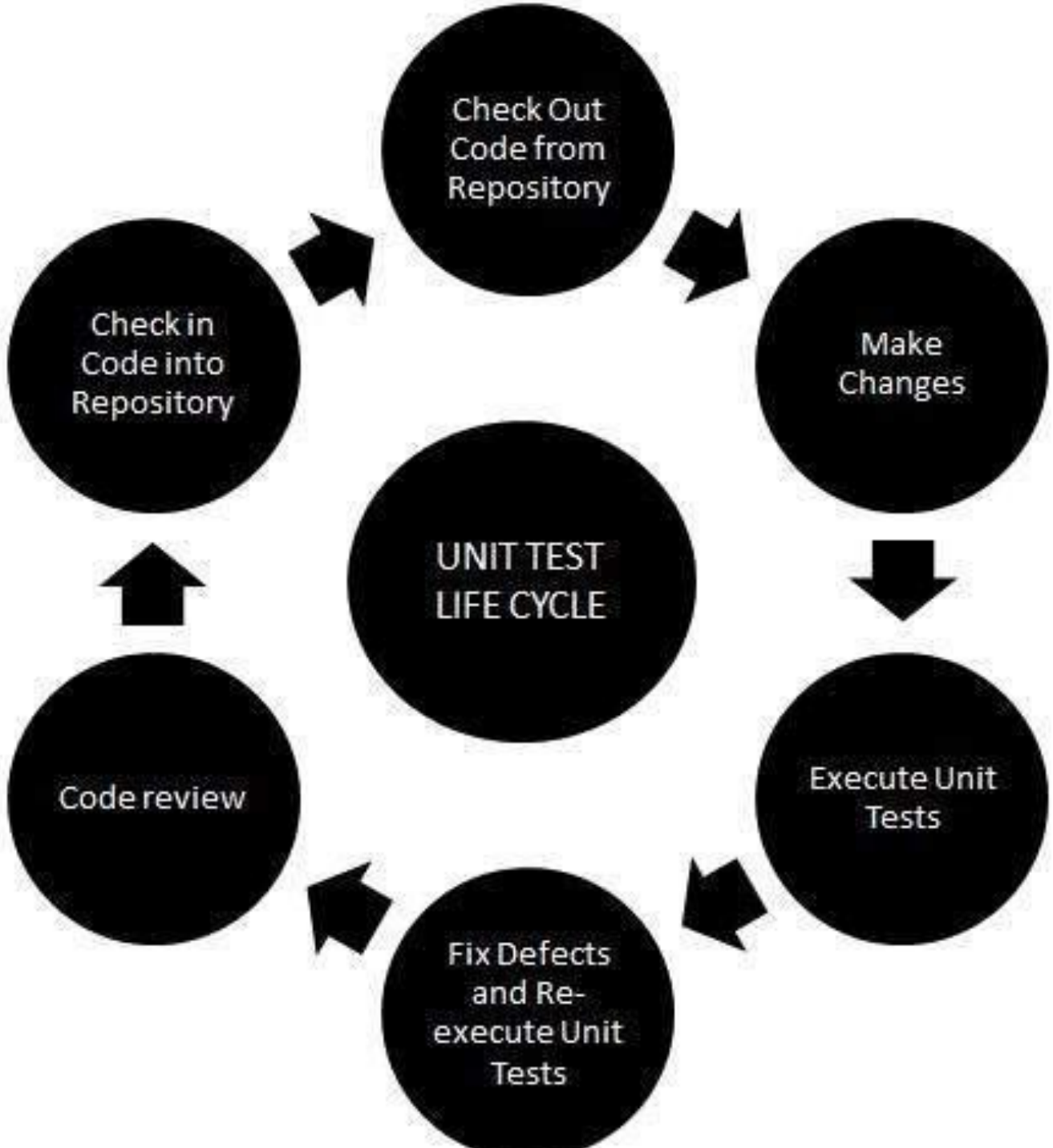
- Unit testing
- Smoke testing
- Regression testing
- Conversion validation
- Integration testing
- UX / usability testing
- System testing
- Load testing

What is Unit Testing?

- Unit testing, a testing technique using which individual modules are tested to determine if there are any issues by the developer himself. It is concerned with functional correctness of the standalone modules.
- The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Unit Testing - Advantages:

- Reduces Defects in the Newly developed features or reduces bugs when changing the existing functionality.
- Reduces Cost of Testing as defects are captured in very early phase.
- Improves design and allows better refactoring of code.
- Unit Tests, when integrated with build gives the quality of the build as well.

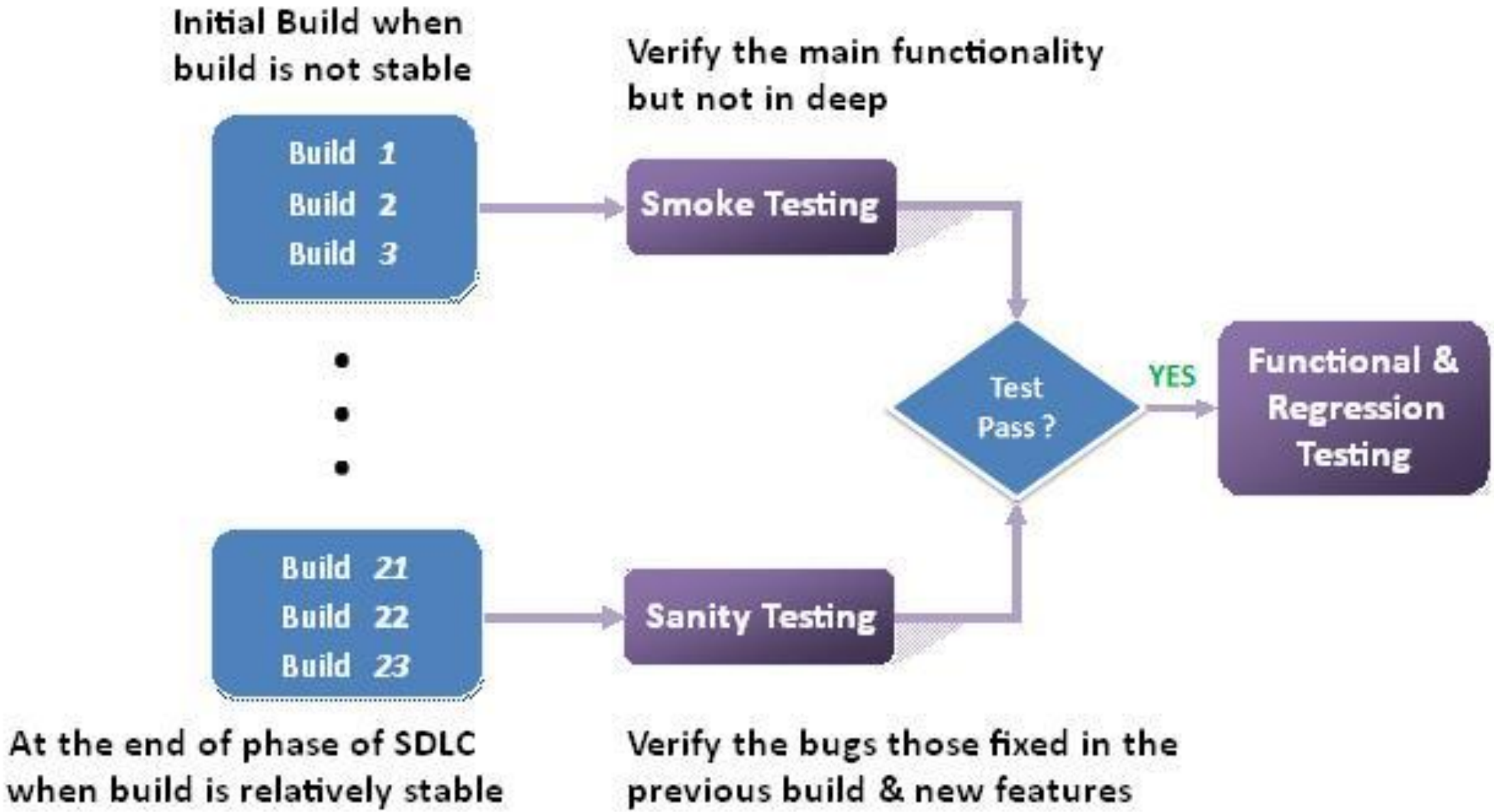

Check Out Code from Repository
Make Changes
Execute Unit Tests
Fix Defects and Re-execute Unit Tests
Code review
Check in Code into Repository
UNIT TEST LIFE CYCLE

Unit Testing Techniques:

- **Black Box Testing -** Using which the user interface, input and output are tested.
- **White Box Testing -** used to test each one of those functions behaviour is tested.
- **Gray Box Testing -** Used to execute tests, risks and assessment methods.

What is Smoke Testing?

- Smoke testing is the initial testing process exercised to check whether the software under test is ready/stable for further testing.
- The term '**Smoke Testing**' is came from the hardware testing, in the hardware testing initial pass is done to check if it did not catch the fire or smoked in the initial switch on.
- Prior to start *Smoke testing* few test cases need to created once to use for smoke testing. These test cases are executed prior to start actual testing to check **critical functionalities of the program is working fine**.


Initial Build when build is not stable
Verify the main functionality but not in deep
Build 1
Build 2
Build 3
Smoke Testing
Test Pass ?
YES
Functional & Regression Testing
Build 21
Build 22
Build 23
Sanity Testing
At the end of phase of SDLC when build is relatively stable
Verify the bugs those fixed in the previous build & new features

Advantages of Smoke testing:

- It helps to find issues introduced in integration of modules.
- It helps to find issues in the early phase of testing.
- It helps to get confidence to tester that fixes in the previous builds not breaking major features (off course, only features exercised by smoke testing).

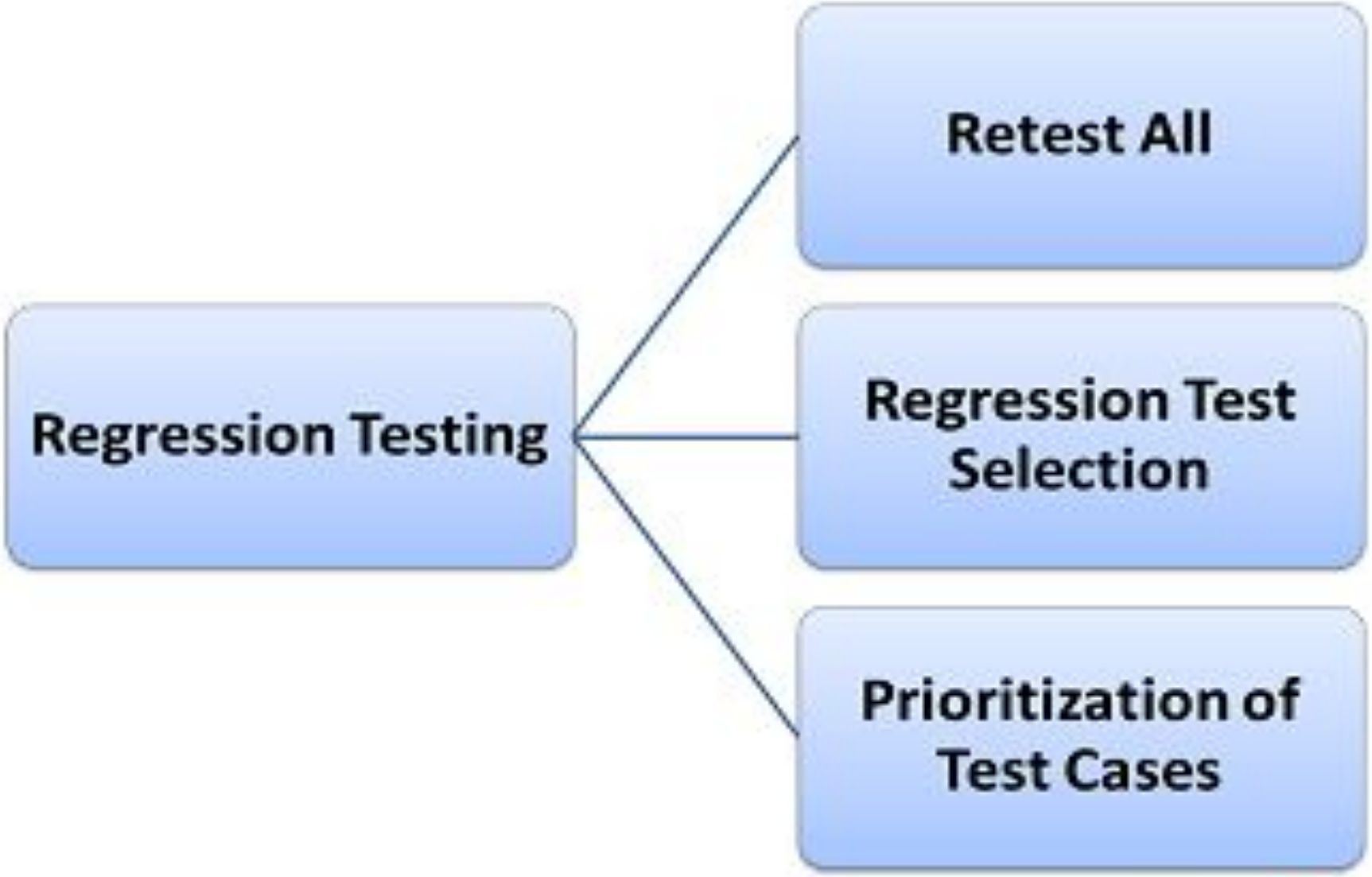
What is Regression Testing?

- Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
- Regression Testing is nothing but full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
- This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that old code still works once the new code changes are done.

Need of Regression Testing

- Change in requirements and code is modified according to the requirement
- New feature is added to the software
- Defect fixing
- Performance issue fix

Regression Testing Techniques



Following are most important tools used for regression testing:

- **Selenium:** This is an open source tool used for automating web applications. Selenium can be used for browser based regression testing.
- **Quick Test Professional (QTP):** HP Quick Test Professional is automated software designed to automate functional and regression test cases. It uses VBScript language for automation. It is a Data driven, Keyword based tool.
- **Rational Functional Tester (RFT):** IBM's rational functional tester is a Java tool used to automate the test cases of software applications. This is primarily used for automating regression test cases and it also integrates with Rational Test Manager.

Following are the major testing problems for doing regression testing:

- With successive regression runs, test suites become fairly large. Due to time and budget constraints, the entire regression test suite cannot be executed
- Minimizing test suite while achieving maximum Test coverage remains a challenge
- Determination of frequency of Regression Tests, i.e., after every modification or every build update or after a bunch of bug fixes, is a challenge.

Conversion validation

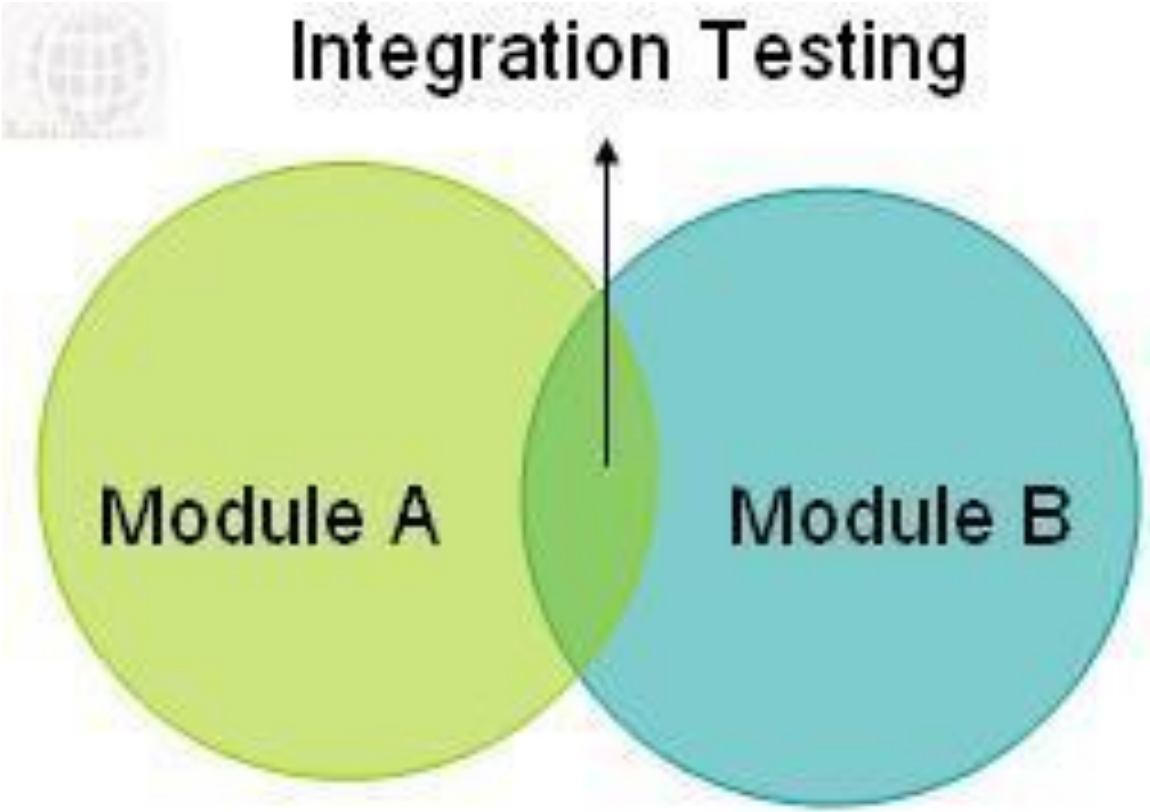
When one “heritage” system is replaced with a newer system, it is common for some historical records or current work in progress to be loaded into the new system at cut-over to production. The goal here is to validate that the records are properly and completely represented in the new system. This requires inspection of both the mapping of the data elements and the translations or manipulations accomplished prior to the load, as well as the related records created by the system during the load process.

What is Integration testing?

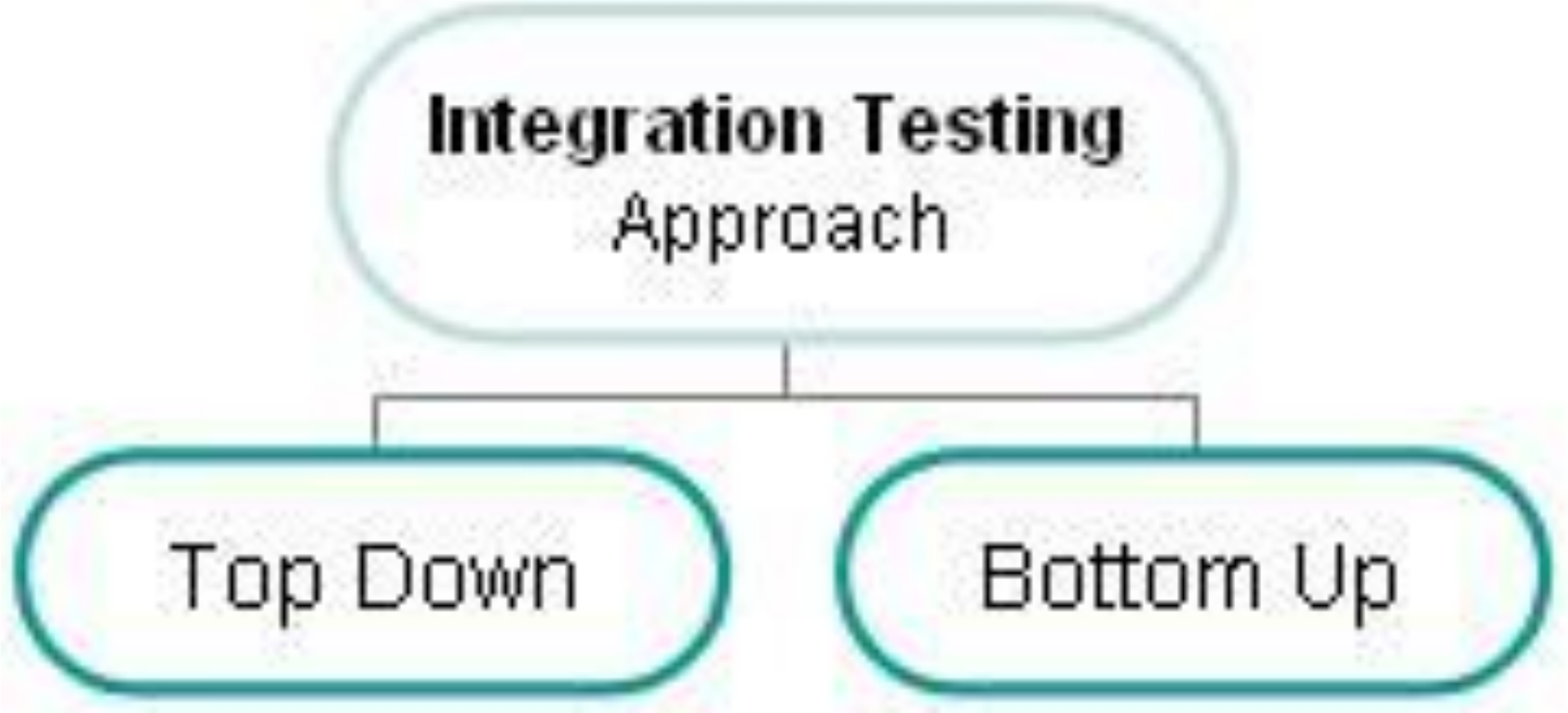
Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.

Integration testing is done by a specific integration tester or test team

Also after integrating two different components togeth er we do the integration testing. As displayed in the image below when two different modules ‘Module A’ and ‘Module B’ are integrated then the integration testing is done.



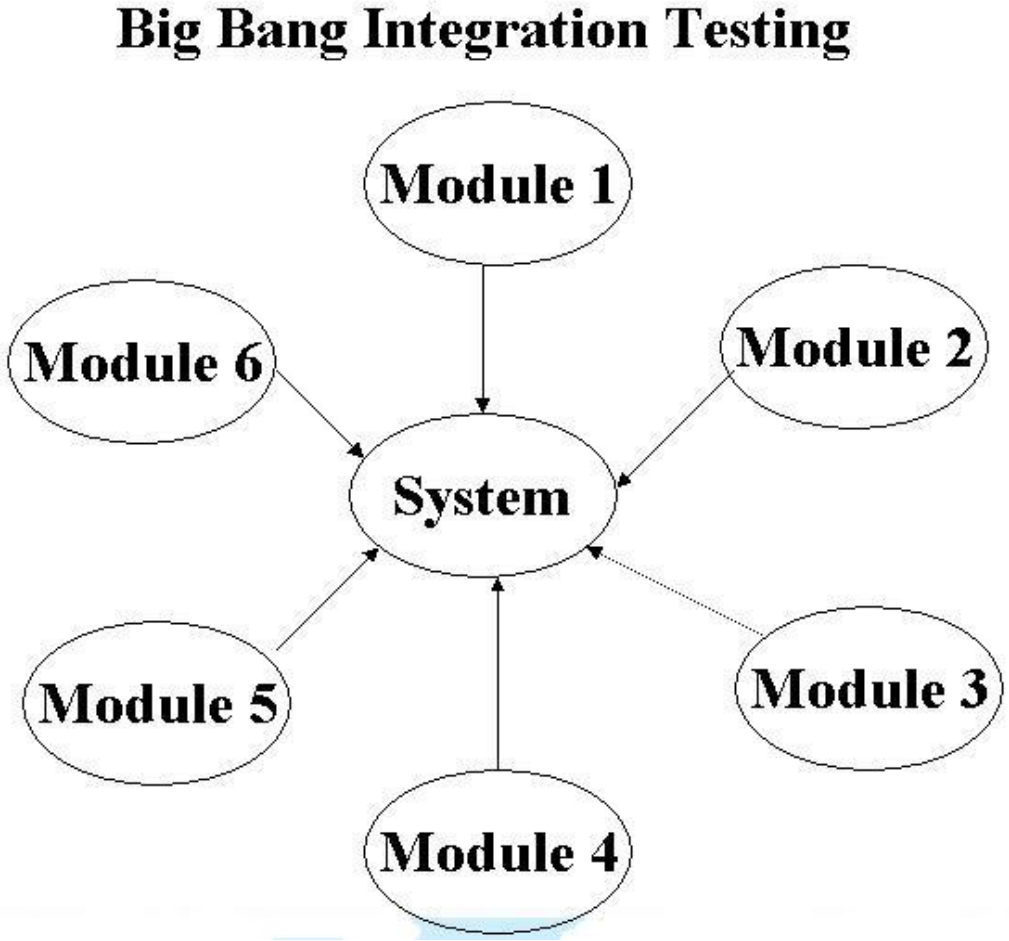
Integration testing follows two approach known as ‘Top Down’ approach and ‘Bottom Up’ approach



Big Bang integration testing

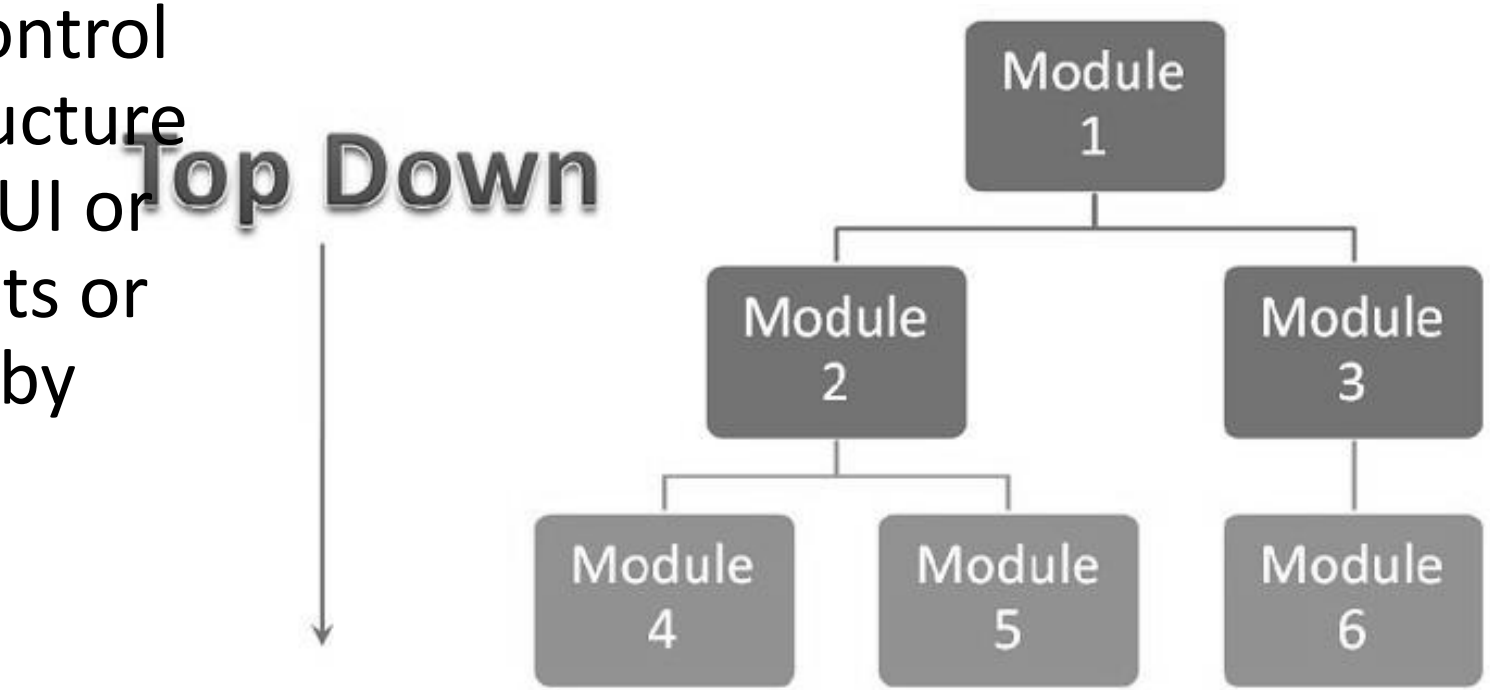
In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole. As per the below image all the modules from ‘Module 1’ to ‘Module 6’ are integrated simultaneously then the testing is carried out.

- **Advantage:** Big Bang testing has the advantage that everything is finished before integration testing starts.
- **Disadvantage:** The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.



Top-down integration testing

Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs.



Advantages of Top-Down approach:

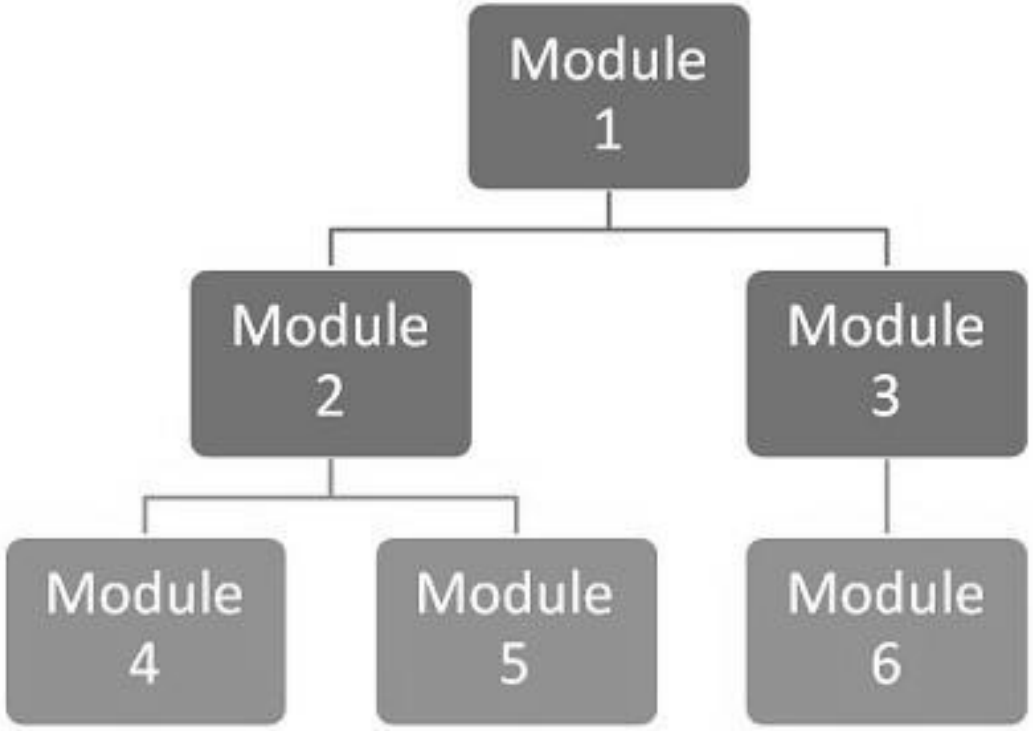
- The tested product is very consistent because the integration testing is basically performed in an environment that almost similar to that of reality
- Stubs can be written with lesser time because when compared to the drivers then Stubs are simpler to author.

Disadvantages of Top-Down approach:

- Basic functionality is tested at the end of cycle

Bottom-up integration testing

Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers.



Advantage of Bottom-Up approach:

- In this approach development and testing can be done together so that the product or application will be efficient and as per the customer specifications.

Disadvantages of Bottom-Up approach:

- We can catch the Key interface defects at the end of cycle
- It is required to create the test drivers for modules at all levels except the top control

- UX / usability testing – Increasingly, we're as concerned about the ability of the user to interact with the system as we are about the ability of the system to correctly operate on the user's instructions. The goal in usability testing is to fine-tune the design of the user experience, in order to maximize delivered value.
- System testing – The notion of system testing has evolved from the early days of simply exercising all of the modules to a more comprehensive end to end test of the integrations with other systems, the correct evolution of a growing set of transaction data records, and validation of access and data security measures. The goal is to ensure that the system, including all software, hardware, network and user actors, works reliably.
- Load testing – As more and more actors make demands on a system whether human users or other systems, performance may be impacted. The goal here is to provide a basis for "tuning" and optimizing delivery of services, whether UX or integration, under a variety of expected demands.

- Survivability testing – Different systems have different availability requirements. Those with high availability or continuity of business operations requirements generally have commensurately more complex designs. The goal here is to exercise key failure modes to ensure that recovery, whether automatic or manual, works as expected.
- User acceptance testing – Decision makers need a basis for accepting delivery of a completed system. The goal here is to exercise not just the system under test but all supporting capabilities, from user training to problem reporting and resolution to managing updates and planned outages. UAT is the final test before the go / no-go decision that determines whether the system can be put into production.
- Parallel testing – This is usually reserved for applications like payroll, where the results of the system under test are compared to the results from the system in production. Like UAT, for applications where it applies, some number of successful parallel cycles are required in order to approve the system for a move to production.

WITHOUT
EXPERIMENTATION
THERE IS NO
INNOVATION.