1ICPC317

AY 2024-25

SDLC Laboratory

Quality Laboratory Manual

Experiment No. 09

To perform various testing using the testing tool: Unit Testing, Integration Testing

Course Instructor –
Mr. Sharanabasava Raddi
ASSISTANT PROFESSOR

Experiment No. 09

Title of Experiment: To perform various testing using testing tool: Unit Testing, Integration

Aim of Experiment: To understand and conduct unit, integration testing using the tool

System Requirements – Win 10 and above OS, 4GB RAM, 2.33 GHz Processor,

Software/s Requirement –

Junit, DataPower

Experiment Objectives:

- Early Detection of Issues
- Improved Code Quality
- Faster Development
- Better Documentation
- Facilitation of Refactoring
- Reduced Time and Cost

Experiment Outcomes:

- Project completion as per the plan i.e. Schedule and Cost
- Delivery of quality product to the customer

Theory:

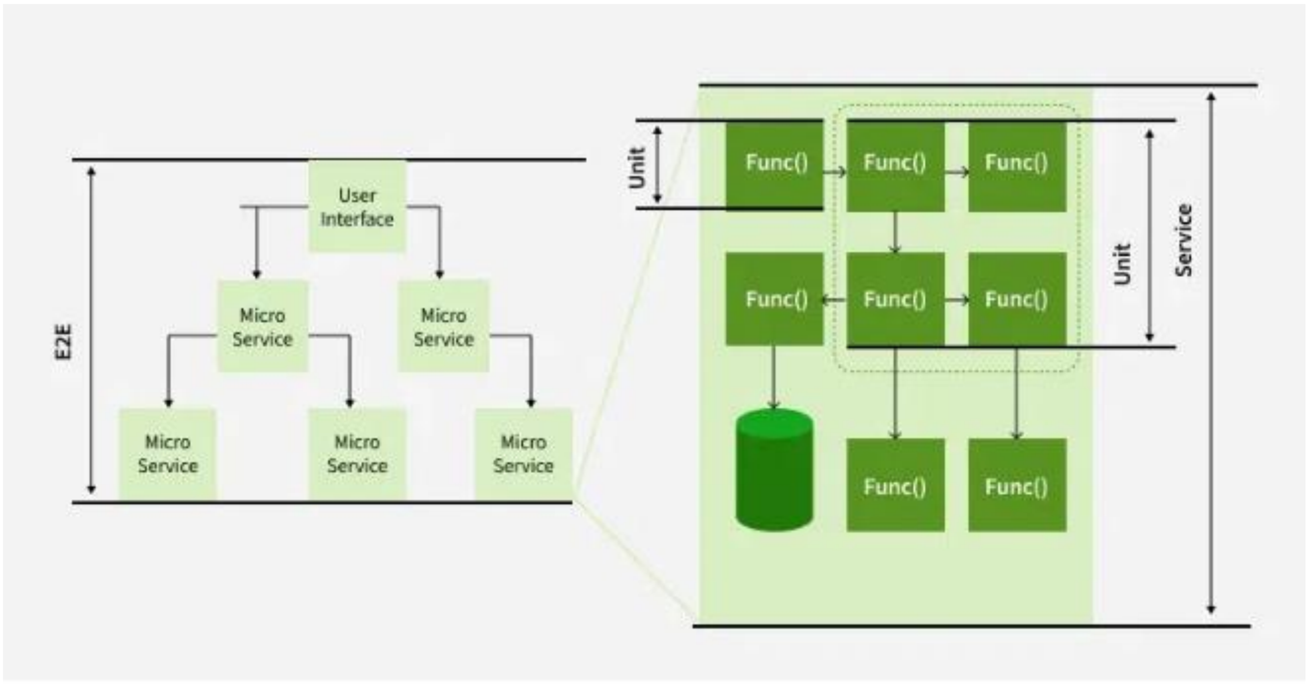
Unit testing: is a type of testing technique in which a single module is tested at a time. It is also known as white box testing. Unit testing checks if a small piece of code in the application is doing what it is supposed to do. In unit testing, a single module is going to be tested at a time, hence the focus of the tester is on the internal design of the application as well. State

Unit testing is a type of testing technique in which a single module is tested at a time. It is also known as white box testing. Unit testing checks if a small piece of code in the application is doing what it is supposed to do. In unit testing, a single module is going to be tested at a time, hence the focus of the tester is on the internal design of the application as well.

To create effective **unit tests**, follow these basic techniques to ensure all scenarios are covered:

- **Logic checks**: Verify if the system performs correct calculations and follows the expected path with valid inputs. Check all possible paths through the code are tested.
- **Boundary checks**: Test how the system handles typical, edge case, and invalid inputs. For example, if an integer between 3 and 7 is expected, check how the system reacts to a 5 (normal), a 3 (edge case), and a 9 (invalid input).
- **Error handling**: Check the system properly handles errors. Does it prompt for a new input, or does it crash when something goes wrong?

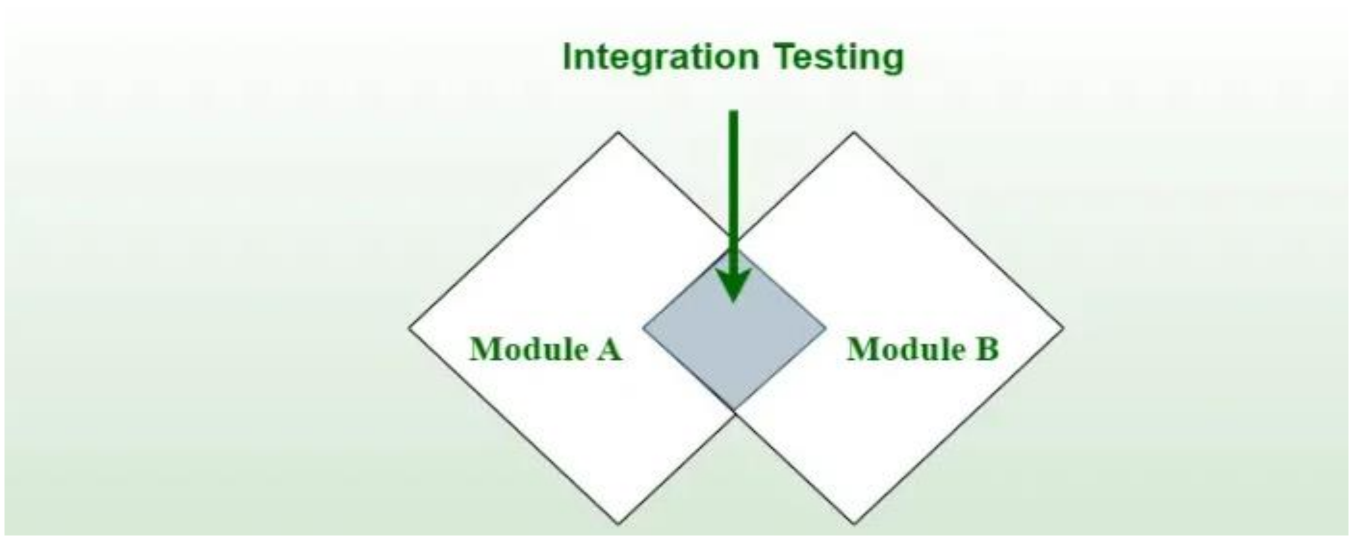
- **Object-oriented checks**: If the code modifies objects, confirm that the object's state is correctly updated after running the code.



Unit Test

Integration Testing: The software integration testing is done to check if communication between the different components of the software is working fine. The primary objective of integration testing is to detect bugs while numerous components of the software are combined. It is conducted once the unit testing has been completed, and just before the software moves on to the system testing phase.

Thus the integration testing helps to identify errors in the early stages of the software development life cycle (SDLC), and minimizes the probability of finding integration issues at the later stages when it becomes costly to fix them. The integration testing is conducted by selecting the unit by unit of a software in a logical sequence.



Types of Integration testing:

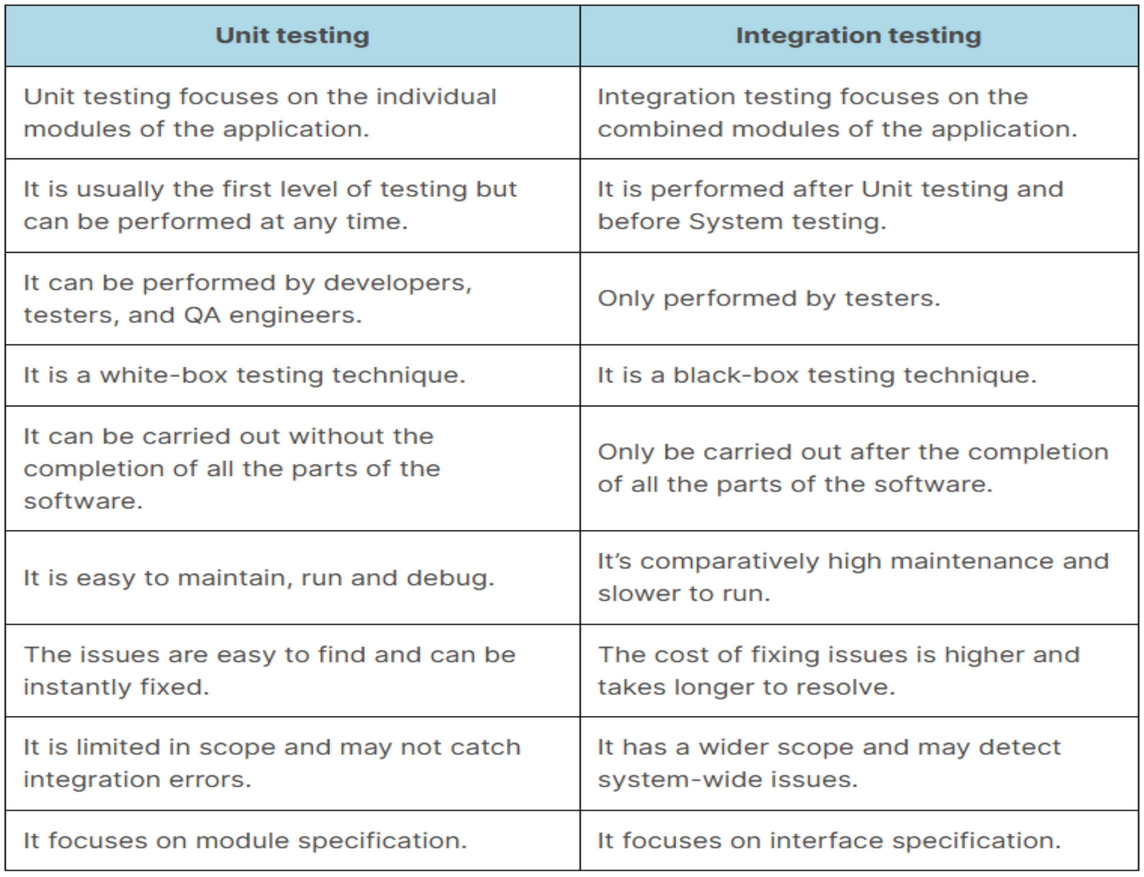
1. **Top Down Integration Testing:** It is a third type of integration testing where the characteristics of the lower level components which are not yet combined are simulated. The testing is conducted from the top to the bottom levels components.

2. **Bottom Up Integration Testing:** It is a second type of integration testing where each module is first verified and then they are incrementally amalgamated with other modules to form a complicated system until the complete software is tested. The lower level modules are validated first followed by the higher level ones.

Observations:

- Clarity: They provide a clear visual representation of system behavior, making it easier for stakeholders to understand.
- Communication: State charts facilitate better communication among team members by providing a common language.
- Documentation: They serve as a valuable documentation tool, capturing the dynamic aspects of a system.

Comparison between Unit and Integration Testing



Unit testing	Integration testing
Unit testing focuses on the individual modules of the application.	Integration testing focuses on the combined modules of the application.
It is usually the first level of testing but can be performed at any time.	It is performed after Unit testing and before System testing.
It can be performed by developers, testers, and QA engineers.	Only performed by testers.
It is a white-box testing technique.	It is a black-box testing technique.
It can be carried out without the completion of all the parts of the software.	Only be carried out after the completion of all the parts of the software.
It is easy to maintain, run and debug.	It's comparatively high maintenance and slower to run.
The issues are easy to find and can be instantly fixed.	The cost of fixing issues is higher and takes longer to resolve.
It is limited in scope and may not catch integration errors.	It has a wider scope and may detect system-wide issues.
It focuses on module specification.	It focuses on interface specification.

Conclusion:
The experiment successfully demonstrated, the way of conducting Unit Testing and Integration Testing and its importance.

Expected Oral Questions:

1. What is Unit Testing?
2. Who is responsible to conduct unit testing?
3. What is integration testing?
4. What are the tools used for Unit Testing and Integration Testing?
5. Specify the different types of Integration Testing

FAQs in Interview:

1. What is the difference between Unit and Integration Testing?
2. What is Stub in integration testing?
3. What is driver in integration testing?
4. What are the difference between unit testing and integration testing?