**UCLM TESTQUA**

Sept. 20, 2025

**Level of Software Testing**

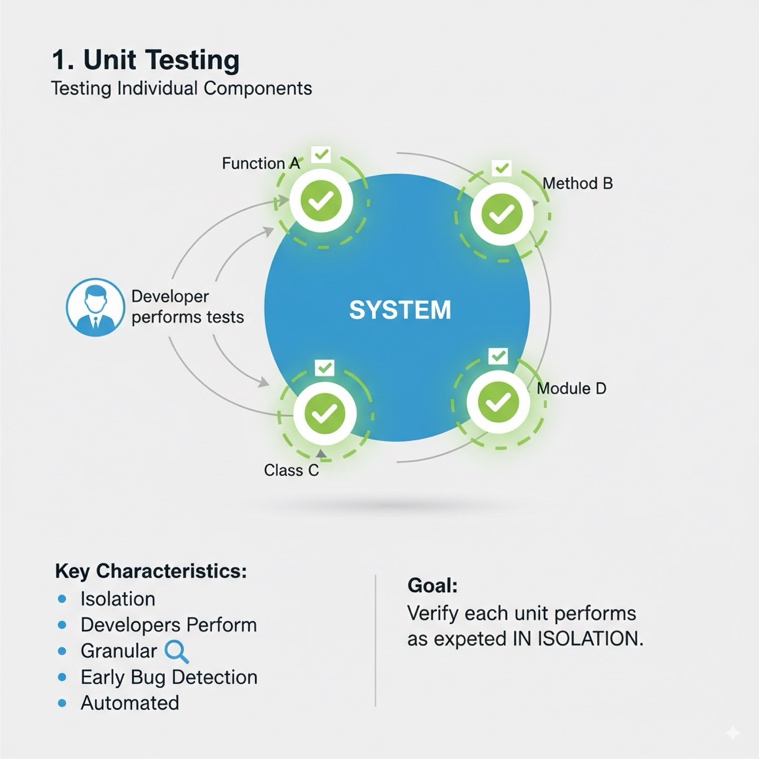
**1. Unit Testing: Testing Individual Components**

Unit testing is the **first level of testing** and focuses on individual components or modules of the software. A "unit" is the smallest testable part of an application, such as a function, method, or class. The primary goal of unit testing is to verify that each unit of the software performs as expected in isolation.

**Key Characteristics:**

* **Isolation:** Units are tested independently, often using mock objects or stubs to simulate dependencies.
* **Developers Perform:** Typically performed by developers themselves as they write the code.
* **Granular:** Focuses on the smallest testable parts.
* **Early Bug Detection:** Catches bugs very early in the development cycle, making them cheaper and easier to fix.
* **Automated:** Often automated using frameworks like JUnit (Java), NUnit (.NET), or Pytest (Python).

**Example:** If you have a function that calculates the sum of two numbers, unit testing would involve testing this function with various inputs (positive, negative, zero) to ensure it always returns the correct sum.



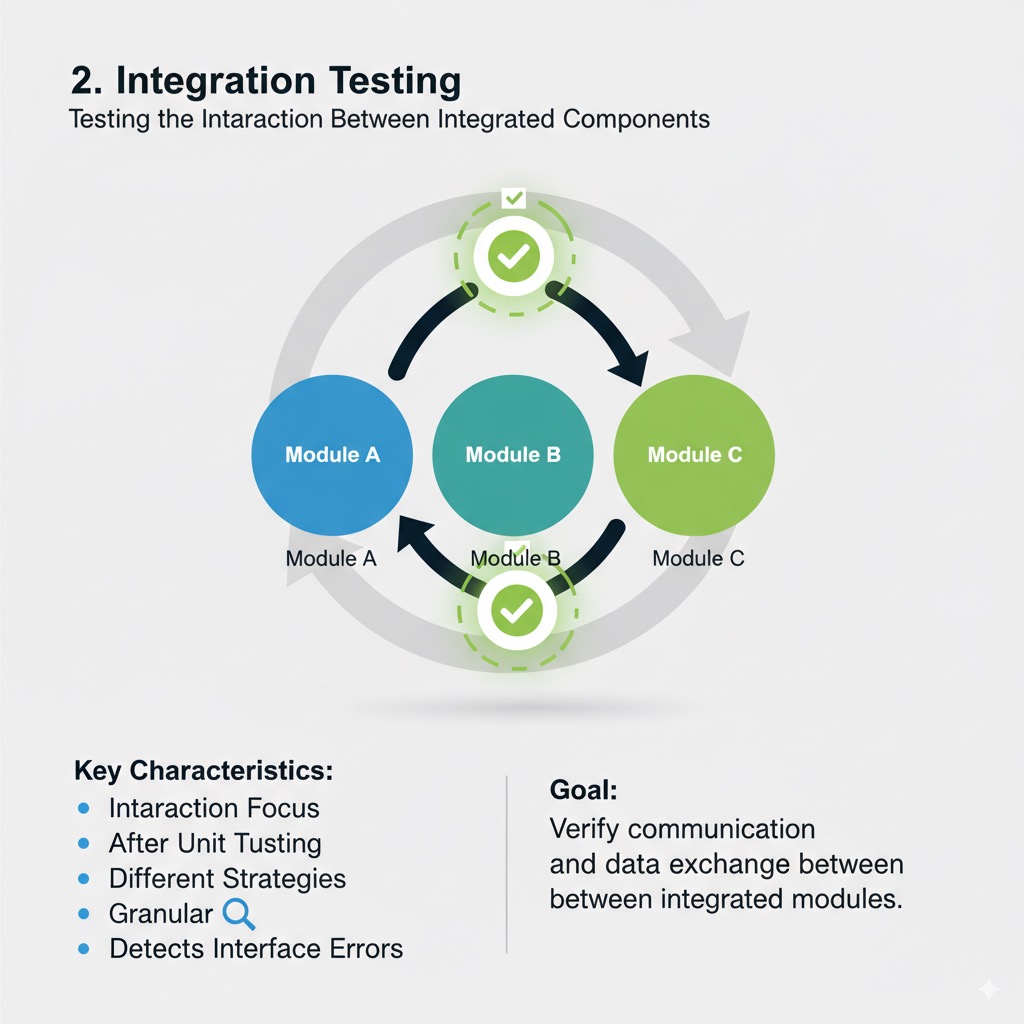
**2. Integration Testing: Testing the Interaction Between Integrated Components**

Integration testing follows unit testing. Its purpose is to test the interactions and interfaces between different units or modules that have been individually tested and then combined. The goal is to ensure that these integrated units work together seamlessly and that data flows correctly between them.

**Key Characteristics:**

* **Interaction Focus:** Verifies the communication and data exchange between integrated modules.
* **After Unit Testing:** Performed after individual units have passed their unit tests.
* **Different Strategies:** Can follow various approaches like "Big Bang," "Top-Down," or "Bottom-Up" integration.
* **Detects Interface Errors:** Aims to uncover defects in the interfaces and communication paths between modules.

**Example:** If you have a module for user authentication and another for user profile management, integration testing would involve ensuring that after a user logs in (authentication module), their profile can be successfully accessed and updated (profile management module).



**3. System Testing: Testing the Complete and Integrated Software**

System testing is a higher-level testing phase where the **complete and integrated software system** is tested as a whole. It evaluates the system's compliance with specified requirements. This stage tests the entire application, including all its modules, components, and their interactions, within an environment that closely resembles the production environment.

**Key Characteristics:**

* **End-to-End Testing:** Covers the entire system, from input to output, across all functionalities.
* **Non-Functional Testing:** Often includes testing non-functional requirements such as performance, security, reliability, and usability.
* **Black Box Testing:** Typically performed from the user's perspective, without knowledge of the internal code structure.
* **Validates Requirements:** Ensures the system meets all functional and non-functional specifications.

**Example:** For an e-commerce website, system testing would involve testing the entire user journey: browsing products, adding to cart, checkout, payment processing, order confirmation, and even checking order history. It would also test how the system performs under heavy load (performance testing) and its security against vulnerabilities.



**4. Acceptance Testing: Validating that the System Meets User Needs**

Acceptance testing is the final level of testing before the software is released to the market or deployed for actual use. Its primary goal is to **validate that the system meets the user's needs and business requirements** and is fit for its intended purpose. It's often performed by the end-users or clients.

**Key Characteristics:**

* **User/Client Focused:** Conducted by actual users or business representatives.
* **Business Requirements Validation:** Ensures the software addresses the defined business problems and user expectations.
* **Sign-Off:** Successful acceptance testing often leads to a formal sign-off, indicating the client's approval for deployment.
* **Types:** Can include User Acceptance Testing (UAT), Business Acceptance Testing (BAT), Operational Acceptance Testing (OAT), and Contract Acceptance Testing (CAT).
* **Real-World Scenarios:** Tests often simulate real-world usage scenarios.

**Example:** Before launching a new banking application, the bank's employees and perhaps a pilot group of customers would perform acceptance testing. They would verify if the app is easy to use, if all banking transactions work as expected, and if it meets all the regulatory and business requirements.

