University of Tripoli – Faculty of Information Technology

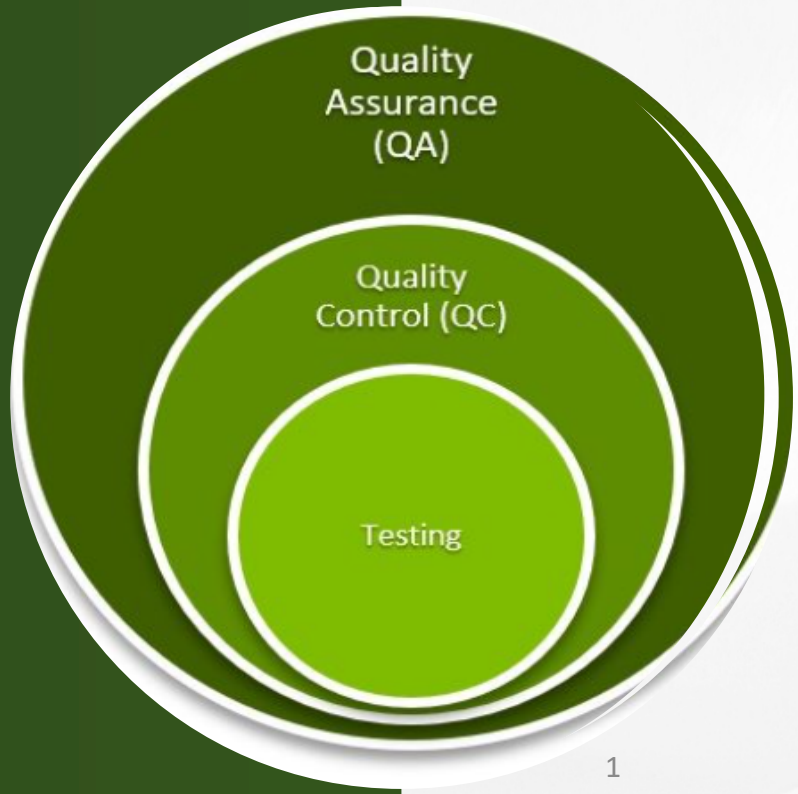
Software Engineering Department

Software Quality Assurance

ITSE421

Spring 2024

By Marwa Solla



Software Quality Assurance and Testing

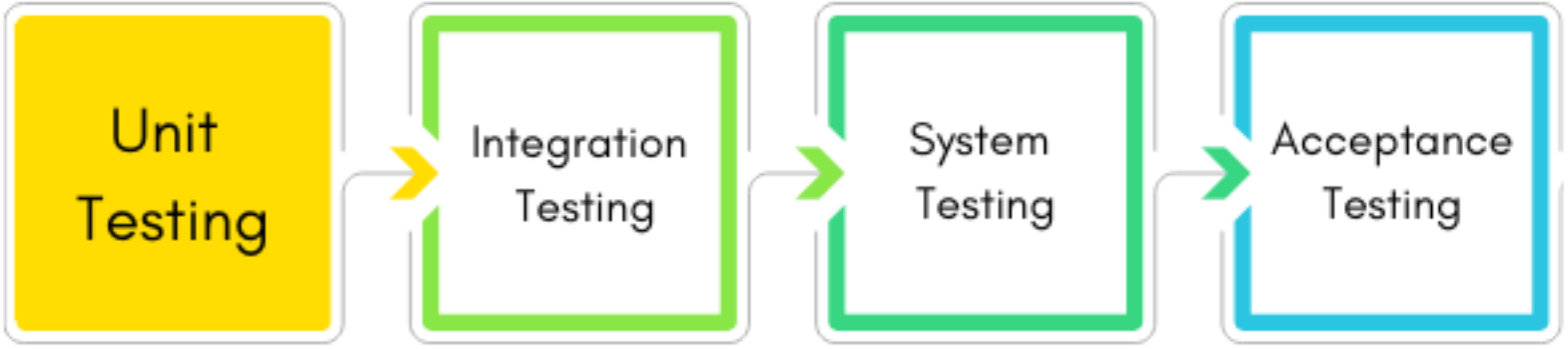
Lecture 5 : Levels of Testing

Marwa Solla
m.solla@uot.edu.ly
Tripoli University - Faculty of Information Technology

Spring 2024

What We Learn In This Lecture

- ❑ Unit or Component testing
- ❑ Integration testing
- ❑ System testing
- ❑ Acceptance testing



Unit Testing

- **Unit Testing** is a is the first level of software testing in which individual software units or components are tested.
- Unit testing (also known as Component or module testing) focuses on components that are separately testable.
- The goal is to ensure that each unit of software code performs as expected. Developers perform unit test during an application's development (coding phase).
- Unit tests isolate and verify the correctness of a section of code. A unit is a single function, method, procedure, module, or object.

Note: Unit is the smallest part that can be tested independent . If the component can't be tested so isn't a Unit.

Unit Testing

- **Unit Testing** :– in unit testing individual component of software tested.
- The purpose of this testing is that each module is working properly It focuses on the smallest unit of software design (done by developer by using sample input and observing its sample output).
- A unit is a single component or module of a software.
- Unit Testing is white box testing technique.
- Can be done by Manual or Automated Testing.
- Various automated unit test software is available such as: **Junit, NUnit , PUPUnit**

Unit Testing

Unit Testing techniques:

- Basis path testing
- Control structure testing
  - ✓ Conditional coverage
  - ✓ Loops Coverage
- Mutation Testing

Objectives of unit (component) testing

Objectives of component testing include

- Reducing risk
- Verifying whether the functional and non–functional behaviors of the component are as designed
- and specified
- Building confidence in the component's quality
- Finding defects in the component
- Preventing defects from escaping to higher test levels

Unit Testing

Test basis

Examples of work products that can be used as a test basis for component testing include:

- Detailed design
- Code
- Data model
- Component specifications

Test objects

Typical test objects for component testing include:

- Components, units or modules
- Code and data structures
- Classes
- Database modules

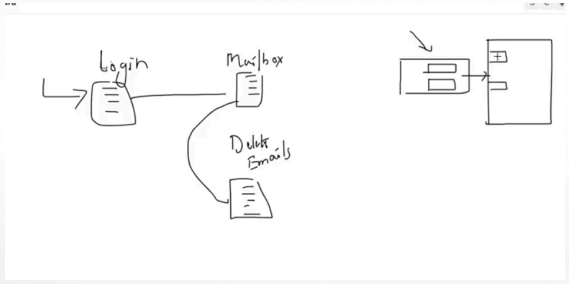
Integration Testing

- Integration testing focuses on interactions between components or systems. Integration testing performed between 2 or more modules.
- Integration testing focuses on checking data communication between multiple modules.
- Integrated Testing is white box testing technique.

Types of integration testing

1) Incremental Integration Testing

2) Non–Incremental Integration Testing



Integration Testing

Incremental Integration Testing: Incrementally adding the modules and testing the data flow between the modules.

- **Top Down–Incremental Integration Testing:**

  Incrementally adding the modules and testing the data flow between the modules. And Ensure the module added is the child of previous module.

- **Bottom Up–Incremental Integration Testing:**

  Incrementally adding the modules and testing the data flow between the modules. And Ensure the module added is the parent of the previous module.

- **Sandwich/Hybrid Approach:**
  Combination of Top–Down & Bottom Up approach is called Sandwich Approach.

Non–Incremental Integration Testing

Adding all the modules in a single shot and test the data flow between modules.

Drawbacks:

1) We might miss data flow between some of the modules.

2) If you find any defect we cant understand the root cause of defect.

Objectives of integration testing

Objectives of integration testing include:

- Reducing risk
- Verifying whether the functional and non–functional behaviors of the interfaces are as designed
- and specified
- Building confidence in the quality of the interfaces
- Finding defects (which may be in the interfaces themselves or within the components or systems)
- Preventing defects from escaping to higher test levels

Integration Testing

There are two different levels of integration testing described in this syllabus, which may be carried out on test objects of varying size as follows:

1) **Component integration testing** focuses on the interactions and interfaces between integrated components. Component integration testing is performed after component testing, and is generally automated.
2) **System integration testing** focuses on the interactions and interfaces between systems, packages, and microservices. System integration testing can also cover interactions with, and interfaces provided by, external organizations (e.g., web services). System integration testing may be done after system testing or in parallel with ongoing system test activities

Integration Testing

Test basis

Examples of work products that can be used as a test basis for integration testing include:

- Software and system design
- Sequence diagrams
- Interface and communication protocol specifications
- Use cases
- Architecture at component or system level
- External interface definitions

Test objects

Typical test objects for integration testing include:

- Subsystems
- Databases
- Interfaces
- APIs
- Microservices

System testing

- **System testing** focuses on the behavior and capabilities of a whole system or product, often considering the end-to-end tasks the system can perform and the non-functional behaviors it exhibits while performing those tasks.
- Testing over all functionality of the application with respective client requirements.
- It is a black box testing technique.
- This testing is conducted by testing team.

System testing

- After completion of component and integration level testing's we start System testing.
- Before conducting system testing we should know the customer requirements.
- System Testing focusses on below aspects.
  - ✓ User Interface Testing (GUI)
  - ✓ Functional Testing
  - ✓ Non–Functional Testing
  - ✓ Usability Testing

System testing

Objectives of system testing include:

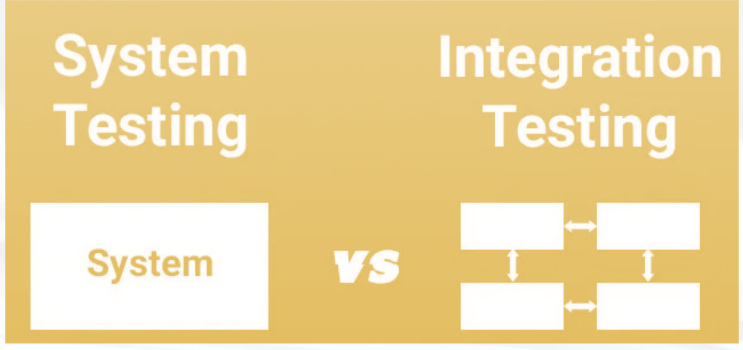
- Reducing risk
- Verifying whether the functional and non–functional behaviors of the system are as designed and specified
- Validating that the system is complete and will work as expected
- Building confidence in the quality of the system as a whole
- It helps in getting maximum bugs before acceptance testing.

System testing Vs Integration testing

System Testing	Integration Testing
Testing the entire software as a whole and understanding if it meets the user requirements	Testing several modules together and understanding how they interface with each other
Both functional and non–functional tests such as performance, security,	Only functional tests Can be performed in different approaches like top–down, bottom–up, big–bang, and more
After Integration Testing	Before System Testing and after Unit Testing

System testing Vs Integration testing

System Testing	Integration Testing
Involves all the software components and checks for bugs	Helps to figure out how different modules communicate with each other
Tests the finished product.	Validates the collection and interface modules.
black–box testing techniques only.	it requires white/grey box testing techniques along with black–box techniques.



Acceptance testing

Acceptance testing

Acceptance testing, like system testing, typically focuses on the behavior and capabilities of a whole system or product.

Objectives of acceptance testing include:

- Establishing confidence in the quality of the system as a whole
- Validating that the system is complete and will work as expected
- Verifying that functional and non-functional behaviors of the system are as specified

Acceptance testing

- Acceptance testing may produce information to assess the system's readiness for deployment and use by the customer (end–user).
- Defects may be found during acceptance testing, but finding defects is often not an objective, and finding a significant number of defects during acceptance testing may in some cases be considered a major project risk.
- Acceptance testing may also satisfy legal or regulatory requirements or standards.
- Common forms of acceptance testing include the following:
  - ✓ **User acceptance testing**
  - ✓ **Operational acceptance testing**
  - ✓ **Contractual and regulatory acceptance testing**
- Types of UAT are Alpha and beta testing.

User acceptance testing (UAT)

- The acceptance testing of the system by users is typically focused on validating the fitness for use of the system by intended users in a real or simulated operational environment.
- The main objective is building confidence that the users can use the system to meet their needs, fulfill requirements, and perform business processes with minimum difficulty, cost, and risk.

Operational acceptance testing (OAT)

The acceptance testing of the system by operations or systems administration staff is usually performed in a (simulated) production environment. The tests focus on operational aspects, and may include:

- Testing of backup and restore
- Installing, uninstalling and upgrading
- User management
- Maintenance tasks
- Data load and migration tasks
- Checks for security vulnerabilities
- Performance testing

Operational acceptance testing (OAT)

The main objective of operational acceptance testing is building confidence that the operators or system administrators can keep the system working properly for the users in the operational environment, even under exceptional or difficult conditions.

Contractual and regulatory acceptance testing

- Contractual acceptance testing is performed against a contract's acceptance criteria for producing custom–developed software.
- Acceptance criteria should be defined when the parties agree to the contract. Contractual acceptance testing is often performed by users or by independent testers.
- Regulatory acceptance testing is performed against any regulations that must be adhered to, such as government, legal, or safety regulations.
- The main objective of contractual and regulatory acceptance testing is building confidence that contractual or regulatory compliance has been achieved.

Alpha and beta testing

Types of UAT

- Alpha Testing
- Beta Testing

Alpha and beta testing

Alpha and Beta testing are types of User Acceptance Testing (UAT). They are typically used by developers of commercial off–the–shelf (COTS) software who want to get feedback from potential or existing users, customers before the software product is put on the market.

Alpha testing

- Alpha site tests" are tests of a new software package that are performed at the developer's site. The customer, by applying the new software to the specific requirements of his organization, tends to examine the package from angles not expected by the testing team.
- The errors identified by alpha site tests are expected to include the errors that only use by a real user can reveal, and thus should be reported to the developer.
- Alpha testing is a type of acceptance testing, which is performed to identify all possible bugs/issues before releasing the product to the end–user.

Alpha testing

- Alpha test is a preliminary software field test carried out by a team of users to find out the bugs that were not found previously by other tests.
- Alpha testing is to simulate a real user environment by carrying out tasks and operations that actual user might perform.

Beta testing

- Beta site tests are much more commonly applied than are alpha site tests. The beta site test process can be described as follows. Once an advanced version of the software package is available, the developer offers it free of charge to one or more potential users.
- The users install the package in their sites (usually called the “beta sites”), with the understanding that they will inform the developer of all the errors revealed during trials or regular usage.
- Participants in beta site testing are often users of previously released packages. Because beta site tests are considered to be a valuable tool, some developers involve hundreds or even thousands of participants in the process.

Beta testing

- Beta testing may come after alpha testing, or may occur without any preceding alpha testing having occurred.

The main advantages of beta site tests are:

■ **Identification of unexpected errors.** Users usually examine software in an entirely different way and, of course, apply it in ways far from those anticipated in the developer's scenarios. Consequently, they reveal errors of a type that professional testers rarely identify.

Beta testing

■ **A wider population in search of errors**. The wide range of participants involved in beta site testing contributes a scope of software usage experience and potential for revealing hidden errors that go beyond those available at the developer's testing site.

■ **Low costs.** As the participants are not paid for their participation or for error information they report, the only costs encountered are the price of the package and its delivery free of charge to the customer. In most cases these costs, including the loss of sales, are relatively low

Beta testing

The main disadvantages of beta site tests are:

- **A lack of systematic testing.** As participants in beta site tests are in no way obligated to prepare orderly reports, they tend to report scattered experience and leave untouched applications as well as segments of those applications.
- **Low quality error reports.** Participants are not professional testers; hence, their error reports are often faulty (some report no errors at all), and it is frequently impossible to reconstruct the error conditions they report.
- **Difficult to reproduce the test environment**. Beta site testing is usually performed in an uncontrolled testing environment, a fact that creates difficulties when attempting to identify the causes of the reported errors.

Benefits of UAT

- If internal User Acceptance Testing executed before delivery to the customer than it ensures that the system is working according to requirements and all the functions are correctly defined.
- It helps to identify that end product works according to expectations.
- To get end product delivered without any bug.
- To get end product delivered in a proper working condition to a user.
- To get satisfied with all the functionalities that an end product should possess.

The end