


Software design and testing

Software design

- Is the process of defining software functions, objects, methods and structure of the programming codes, so that it matches the desired expectations

 Software testing is the next step wherein the software is put through trials to ensure that it matches expected requirements and is error free.

Software design



the design phase deals with transforming the customer requirements into a form implementable using a programming language. It is divided into 3 phases:

- Interface design

- is the specification of the interaction between a system and its environment

- Architectural design

- Is the specification of the major components of a system.

- Detailed design

- is the specification of the internal elements of all major system components, their properties, relationships and processing. It decompose the system components into units


Software Testing (ST)

- 📄 is a methods to check whether the actual software product matches expected requirements and ensure that it is error free.
- 📄 testing is sometimes call it white box and black box testing.
- 📄 Simply testing means the verification of application under test (AUT)

Remark for system testing (Waterfall Model)

- 📄 **Alpha-testing:** is the system testing performed by the development team
- 📄 **Beta-testing:** is the system testing performed by a friendly set of customers
- 📄 **Acceptance testing:** after the software has been delivered, the customer performed the acceptance testing to determine whether to accept the delivered software or to reject it

Remark

 The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability.

Types of ST

Functional testing

- Unit testing
- Integration testing
- User acceptance testing

Non-functional testing

- Performance
- Usability
- scalability

Maintenance testing

When software Bugs (errors/fault/failure/problem) occurs?


- ☞ The software does not do something that the specification says it should do
- ☞ The software does something that the specification says it should not do
- ☞ The software is difficult to understand, hard to use or slow
- ☞ Note: Most bugs are not because of mistakes in the code BUT may be in terms of Specification (55%), design (25%), code (15%), others (5%)


Remark

- **Mistake** – a human action that produces an incorrect result.
- **Fault [or Defect]** – an incorrect step, process, or data definition in a program.
- **Failure** – the inability of a system or component to perform its required function within the specified performance requirement.
- **Error** – the difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.
- **Specification** – a document that specifies in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristic of a system or component, and often the procedures for determining whether these provisions have been satisfied.

ST Process (Cont.)

 **Software testing can be divided into two steps:**

 **Verification:** it refers to the set of tasks that ensure that software correctly implements a specific function.

 **2. Validation:** it refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

 **Verification:** “Are we building the product right?”

Validation: “Are we building the right product?”

Verification Example

Verification: Are we building the product right?

Through verification, we make sure the product behaves the way we want it to. For example, on the left in Figure 1, there was a problem because the specification said that players should collect \$200 if they land on or pass Go. Apparently a programmer implemented this requirement as if the player had to pass Go to collect. A test case in which the player landed on Go revealed this error.

Verification

Are we building the product right?



"I landed on "Go" but didn't get my \$200!"

Validation

Are we building the right product?



"I know this game has money and players and "Go" – but this is not the game I wanted."

Figure 1: Verification vs. Validation

Validation Example

Validation: Are we building the right product?

Through validation, we check to make sure that somewhere in the process a mistake hasn't been made such that the product build is not what the customer asked for; validation always involves comparison against requirements. For example, on the right in Figure 1, the customer specified requirements for the Monopoly game – but the programmer delivered the game of Life. Maybe the programmer thought he or she

“knew better” than the customer that the game of Life was more fun than Monopoly and wanted to “delight” the customer with something more fun than the specifications stated. This example may seem exaggerated – but as programmers we can miss the mark by that much if we don't listen well enough or don't pay attention to details – or if we second guess what the customer says and think we know better how to solve the customer's problems.

Verification

Are we building the product right?



“I landed on “Go” but didn't get my \$200!”

Validation


Are we building the right product?



“I know this game has money and players and “Go” – but this is not the game I wanted.”

Figure 1: Verification vs. Validation



What are different types of software testing?

 **Manual Testing:** Manual testing includes testing software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug.


 There are different strategies (stages) for manual testing such as

- unit testing,
- integration testing,
- system testing, and
- user acceptance testing.

What are the different types of software testing? (Cont.)

-  **Automation Testing:** Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves the automation of a manual process.
-  Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

Manual Testing Strategies


 Unit testing: this testing is followed by the programmer to test the unit of the program.

 Integration testing

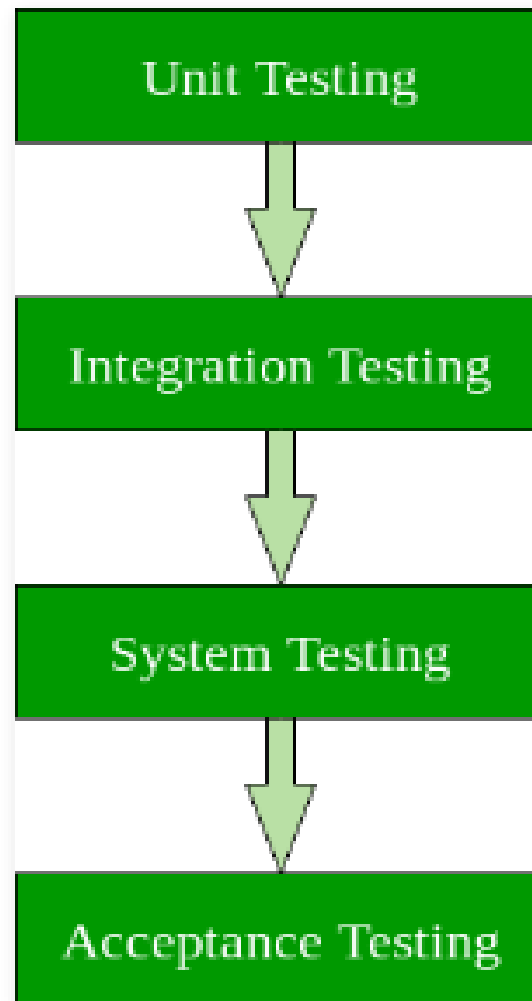
- It focuses on the construction and design of the software.

 System testing

- Your software is compiled as a whole and then tested as a whole

 Acceptance testing: in this level a system is tested for acceptability

Cont.



What are different levels of software testing?

1. **Unit Testing:** A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
2. **Integration Testing:** A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
3. **System Testing:** A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.
4. **Acceptance Testing:** A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

Cont.

Acceptance testing is conducted by business owners, the purpose of acceptance testing is to test whether the system does in fact, meet their business requirements

Regression Testing is the testing of software after changes has been made; this testing is done to make sure that the reliability of each software release, testing after changes has been made to ensure that changes did not introduce any new errors into the system

Alpha Testing Usually in the existence of the developer at the developer's site will be done.

Beta Testing Done at the customer's site with no developer in site.

Functional Testing is done for a finished application; this testing is to verify that it provides all of the behaviors required of it

Alpha and Beta testing

- Alpha testing is an in-house testing process performed within the organization by the developer or tester
- Beta testing is deployed on the customer site for getting tested by the intended users or clients in the real enviro\

\
\\.


What are the different techniques of Software Testing?



Black Box Testing (specification-based

testing): is a ST technique where the functionality of the system is tested without comprehending or looking into the internal structure

white box testing or structural testing

 **White-Box Testing:** The technique of testing in which the tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing or structural testing

White Box Testing Tools

Below is a list of top white box testing tools.

- [Parasoft Jtest](#)
- [EclEmma](#)
- [NUnit](#)
- [PyUnit](#)
- [HTMLUnit](#)
- [CppUnit](#)

Example: NUnit is a unit-testing framework for all .Net languages.

Cont.

Black Box Testing

Internal workings of an application are not required.

Also known as closed box/data-driven testing.

End users, testers, and developers.

This can only be done by a trial and error method.

White Box Testing


Knowledge of the internal workings is a must.


Also known as clear box/structural testing.

Normally done by testers and developers.

Data domains and internal boundaries can be better tested.

Gray Box Testing

 **Grey Box Testing** or Gray box testing is a software testing technique to test a software product or application with partial knowledge of internal structure of the application.

 The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications.

Gray Box Testing (Cont.)

Gray Box Testing is a software testing method, which is a combination of both White Box Testing and Black Box Testing method.

- In White Box testing internal structure (code) is known
- In Black Box testing internal structure (code) is unknown
- In Grey Box Testing internal structure (code) is partially known



Techniques used for Grey box Testing are

Techniques used for Grey box Testing are-

- **Matrix Testing:** This testing technique involves defining all the variables that exist in their programs.
- **Regression Testing:** To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within a firewall.
- **Orthogonal Array Testing or OAT:** It provides maximum code coverage with minimum test cases.
- **Pattern Testing:** This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened

Steps to perform Grey box Testing are

- **Step 1:** Identify inputs
- **Step 2:** Identify the outputs
- **Step 3:** Identify the major paths
- **Step 4:** Identify Subfunctions
- **Step 5:** Develop inputs for Subfunctions
- **Step 6:** Develop outputs for Subfunctions
- **Step 7:** Execute test case for Subfunctions
- **Step 8:** Verify the correct result for Subfunctions
- **Step 9:** Repeat steps 4 & 8 for other Subfunctions
- **Step 10:** Repeat steps 7 & 8 for other Subfunctions

Validation: Are we building the right product?

Through validation, we check to make sure that somewhere in the process a mistake hasn't been made such that the product build is not what the customer asked for; validation always involves comparison against requirements. For example, on the right in Figure 1, the customer specified requirements for the Monopoly game – but the programmer delivered the game of Life. Maybe the programmer thought he or she