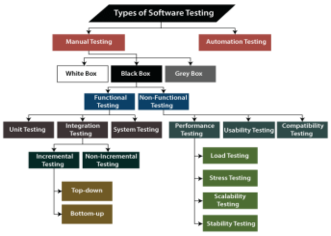
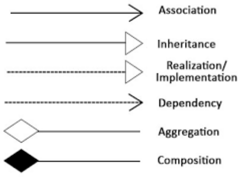
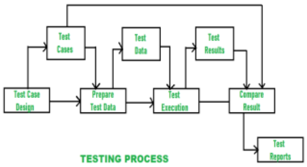




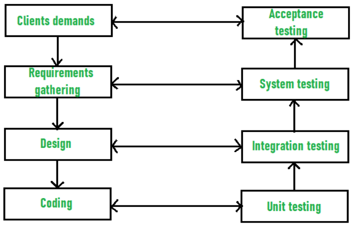


TESTING PROCESS

Software Design and Testing

Verification & Validation

Chapter 3,7



LEVELS OF TESTING

Verification Requirements

- All the requirements gathered from the user's viewpoint are verified.
- An acceptance criterion is prepared for that, which defines the goals and requirements of the proposed system and acceptance limits for each of the goals and requirements.
- Acceptance criterion is most important in case of real-time systems where performance is a critical issue in certain events.
- So it must be defined by the designers of the system and should not be overlooked, as they can create problems while testing the system.
- The tester works in parallel by performing the following two tasks:
  - The tester reviews the acceptance criteria in terms of its completeness, clarity, and test-ability so that necessary resources can be planned.
  - The tester prepares the Acceptance Test Plan which is referred at the time of Acceptance Testing.

Verification of Objective

- After gathering the requirements , objectives are prepared in a document called SRS.
- Here two parallel activities are performed by the tester:
  - Verifying all the objectives mentioned in SRS to ensure that the user's needs are properly understood before proceeding further.
  - Preparing System Test Plan which is based on SRS to be referenced at the time of System Testing.
- In the verification of requirements and objectives , the tester must consider **both functional and non-functional** requirements

How to Verify Requirement and Objective

- Both requirements and objectives has a potential of detecting bugs.
- Testers use SRS for verification of objectives.
- So one characteristic of a good SRS is it can be verified BUT:
  - An SRS can be verified, if and only if, every requirement stated there can be verified.
  - A requirement can be verified, if and only if, there is some procedure to check that the s/w meets its requirements.
- So specify the requirements in a quantification manner means ambiguous statements or language like 'good quality', 'usually', 'may happen' should be avoided.
- Instead quantified specifications should be provided. E.g.
  - Module x produce o/p within 15 sec of its execution OR
  - The o/p should be displayed like this: TRACK A's speed is 'x

Measures For Verify The Requirement in SRS

- Correctness
- Unambiguous
- Consistent
- Completeness
- Updation
- Traceability
  - Backward Traceability
  - Forward Traceability

Characteristics of Good SRS

- SRS should be **accurate, complete, efficient, and of high quality,** so that it does not affect the entire project plan.
- An SRS is **said to be of high quality when** the developer and user **easily understand** the prepared document.
- Characteristics of a Good SRS:

Correct

SRS is correct when all user requirements are stated in the requirements document.

Note that there is no specified tool or procedure to assure the correctness of SRS.

Unambiguous

SRS is unambiguous when every stated requirement has only one interpretation.

Complete

SRS is complete when the requirements clearly define what the software is required to do.

Characteristics of Good SRS

Ranked for Importance/Stability

All requirements are not equally important, hence each requirement is identified to make differences among other requirements.

Stability implies the probability of changes in the requirement in future.

Modifiable

The requirements of the user can change, hence requirements document should be created in such a manner that those changes can be modified easily.

Traceable

SRS is traceable when the source of each requirement is clear and facilitates the reference of each requirement in future.

Verifiable

SRS is verifiable when the specified requirements can be verified with a cost-effective process to check whether the final software meets those requirements.

Consistent

SRS is consistent when the subsets of individual requirements defined do not conflict with each other.

Verification of High Level Design

- The architecture and design are documented in SDD (S/W Design Document).
- The tester requires to perform two parallel activities:
  - Tester needs to verify HLD as its consisting of number of sub-systems or components and functionality of each component should be verified.
  - This is a macro level representation and no details of the individual modules are available so its difficult to design interface for the system to interact with the outside world.
- Every requirement in SRS should map the design to verify that all the components and interfaces are in tune with the requirements of the user.
- Tester also prepares a Function Test Plan, Integration Test Plan to be referenced in Function Testing and in Integration Testing.

How to Verify of High Level Design?

- Here there is highest possibility of finding bugs.
- So a formal specification of design is required known as SDD (S/W Design Document) according to the standard provided by IEEE.
- If bug is not detected in HLD, its fixing cost increases with every phase so verification of HLD is crucial.
- This design is divided in three parts:
  - Data Design
  - Architectural Design
  - Interface Design

Verification of Data Design

- The points to be considered here:
  - Check whether the sizes of data structure have been estimated appropriately.
  - Check the provisions of overflow in a data structure.
  - Check the consistency of data formats with the requirements.
  - Check whether data usage is consistent with its declaration.
  - Check the relationships among data objects in data dictionary.
  - Check the consistency of databases and data warehouses with the requirements specified in SRS.

Verification of Architecture Design

- The points to be considered are:
  - Check that every functional requirement in SRS has been taken care in this design.
  - Check whether all exceptions handling conditions have been taken care.
  - Verify the process of transformation mapping and transaction mapping used for transition from the requirement model to architectural design.
  - Check the functionality of each module according to the requirements specified.
  - Check the inter-dependence and interface between the modules.
  - Coupling and Module Cohesion.

Verification of User-Interface Design

- Check all the interfaces between modules according to architecture design.
- Check all the interfaces between software and other non-human producer and consumer of information.
- Check all the interfaces between human and computer.
- Check all the above interfaces for their consistency.
- Check that the response time for all the interfaces are within required ranges.
- Help Facility verify:
  - The representation of help in its desired manner.
  - The user returns to the normal interaction from help.

Verification of User-Interface Design

- For error messages and warnings , verify:
  - Whether the message clarifies the problem.
  - Whether the message provides constructive advice for recovering from the error.
- For types command interaction, check the mapping between every menu option and their corresponding commands

Verification of Low Level Design

- LLD is a detailed design of modules and data are prepared such that an operational s/w is ready.
- The details of each module is prepared in their separate SRS and SDD.
- Testers need to perform parallel activities in this phase:
  - The tester verifies the LLD. The details and logic of each module is verified such that high level design and low-level design abstractions are considered.
  - The tester also prepares the Unit Test Plan which will be preferred at the time of Unit Testing.

How to Verify Low Level Design?

- A last preceding phase where internal details of each design entity are described:
  - Verify the SRS of each module.
  - Verify the SDD of each module.
  - In LLD, data structures, interfaces and algorithms are represented by design notations; so verify the consistency of every item with their design notations.
- Organizations can build a two-way traceability matrix between the SRS and design (both HLD and LLD) such that at the time of verification of the design each requirement in SRS is verified.

How to Verify Code?

- Check that every design specification in HLD and LLD has been coded using traceability matrix.
- Examine the code against a language specification checklist.
- Verify every statement, control structure, loop, and logic.
- Misunderstood or incorrect Arithmetic precedence.
- Mixed mode operations.
- Incorrect initialization.
- Precision Inaccuracy.
- Incorrect symbolic representation of an expression.
- Different data types.
- Improper or nonexistent loop termination.
- Failure to exit.

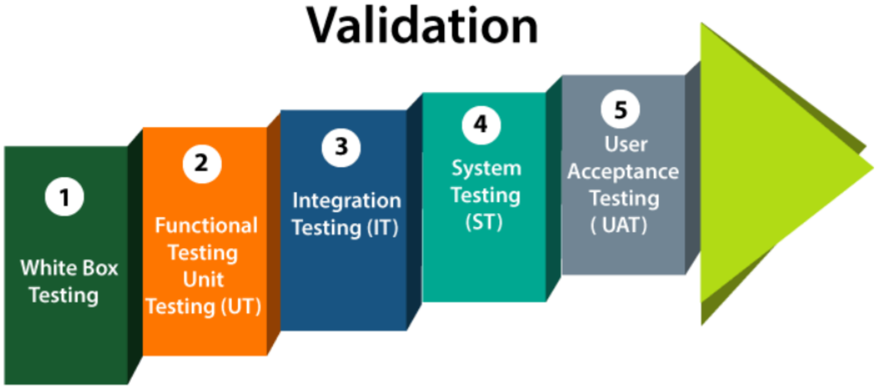
How to Verify Code?

- Two types of techniques are used to verify the coding:
- **Static testing techniques** :No actual execution. Only static analysis of the code or type of conceptual analysis of the code
- **Dynamic testing techniques :** Complementary to static testing. It executes the code on some test data

Unit Verification

- Verification of coding cannot be done for the whole system.
- So verification of coding means verification of code of modules by their developers.
- Points to be considered:
  - Interfaces are verified to ensure that information properly flows in and out of the program unit under test.
  - The local data structure is verified to maintain data integrity.
  - Boundary conditions are checked to verify that the module is working fine on boundaries.
  - All independent paths through the control structure are exercised at least once.
  - All error handling paths are tested.
- Unit verification is largely **white-box oriented.**

Validation



1 White Box Testing

2 Functional Testing Unit Testing (UT)

3 Integration Testing (IT)

4 System Testing (ST)

5 User Acceptance Testing (UAT)

Validation

- Validation is a set of activities that ensures the s/w under consideration has been built right and is traceable to customer's requirements.
- Validation testing is performed after the coding is over.
- **Need for validation :**
  - Developing tests that will determine whether the product satisfies the users' requirements, as stated in the requirement specification.
  - Developing tests that will determine whether the product's actual behavior matches the desired behavior, as described in the functional design specification.
  - The bugs (last chance), which are still existing in the software after coding need to be uncovered else it will move to final product.
  - Validation enhances the quality of software.

Validation Activities

- **Validation Test Plan**
  - Acceptance Test Plan
  - System Test Plan
  - Function Test Plan
  - Integration Test Plan
  - Unit Test Plan
- **Validation Test Execution**
  - Unit Validation Testing
  - Integration Testing
  - Function Testing
  - System Testing
  - Acceptance Testing
  - Installation Testing

Validation Activities

- **Validation Test Plan**
  - Starts as soon as the first o/p of SDLC, SRS, is prepared
  - To prepare validation test plan:
    - Testers must understand the current SDLC phase by studing the relevant documents in the corresponding SDLC phase.
    - With the understanding of SDLC phase and related documents, test plan is to be created to use in validation testing including a sequence of test cases.
  - **Acceptance Test Plan**: prepared in requirement phase according to the acceptance criteria prepared from the user feedback. Used in acceptance testing.
  - **System Test Plan**: prepared to verify the objectives set in SRS. Designed keeping in view how a complete integrated system will behave in different conditions. Used in integration testing

Validation Activities

- **Validation Test Plan**
  - **Function Test Plan**: prepared in HLD phase, in order to test all the interfaces and every functionality. Used in Function Testing.
  - **Integration Test Plan**: prepared to validate integration of all the modules such that all their independences are checked. Used in integration testing.
  - **Unit Test Plan**: prepared in LLD phase. Consists of test plan of every module in the system separately to test functionality related to individual unit can be tested (structured → module, Object oriented system → module/class/package). Used at the time of Unit Testing .

Validation Activities

- **Validation Test Execution**
  - **Unit Validation Testing:** the process of testing individual components of a system along with its all interfaces and functionalities.
    - A unit/module must be validated before integrating it with other modules.
    - First activity after coding phase.
  - **Integration Testing:** process of combining and testing multiple components or modules together.
    - To uncover the bugs that are present when unit tested modules are integrated.
  - **System Testing**: does not aim to test individual function but the system as a whole on various grounds where bugs may occur. E.g. if the system fails in some conditions, how does it recover?

Validation Activities

- **Validation Test Execution**
  - **Acceptance Testing:** acceptance criteria is developed in the requirement phase and system can be tested against that criteria contracted with the customer when the system is ready.
  - **Installation Testing**: Once testing team gives OK for producing the s/w, it is places into an operational status where it is installed.
    - The installation testing does not test the system, but it tests the process of making the s/w system operational.