SWE 4603

Software Testing and Quality Assurance

Lecture 2

Prepared By Maliha Noushin Raida, Lecturer, CSE
Islamic University of Technology

Lesson Outcome

- Verification and Validation
- Various verification activities
- Various validation activities
- How to perform verification at each stage of SDLC
- Various validation test plans

Week 2 On:

- Chapter 3: Verification and Validation

Release Versioning

Versions and Releases are denoted using a quadruplet of integers to indicate Major, Minor, and Revision (Rev); Build numbers are for internal tracking and verification of the software build process and will not be visible to customers are part of the software version number

<MAJOR>. <MINOR>. <REVISION>.<BUILD>

Example: “**2.4.7.1333**”

The version above is reference as version 2.4.7, Where: **Major** release is 2, **Minor** release is 4, **Rev release** is 7 and **Build** is not used except for internal tracking/deployments

Major Release

- A Major Release is a full product release of the software.
- It generally contains new customer-facing functionality and represents a significant change to the code base comprising the software product or family of products, or is used to represent a significant marketing change or direction in the product.

Scope:

- Any change to the code base that prevents backwards compatibility (e.g. Addition or Removal of features, changes in the DB schema or API commands).
- Any new functionality that is customer facing.
- Any large marketing push that accompanies the product and redirects the product.

Frequency:

- Market driven

Audience:

- New customers.
- Existing customers with qualifying contracts
- Existing customers desiring to upgrade to the new feature set

Minor Release

- A Minor Release of the software may be comprised a rollup of several branched releases, enhancements/extensions to existing features or interfaces driven by internal or external requirements.
- external requirements could be driven by enhancements to meet new sales area , internal requirements could be enhancements aligned to a new marketing push.

Scope:

- Minor enhancements and features that do not affect compatibility with its associated major or current minor releases. New features or functionality that does minimally affect the interfaces
- Addition of new features or functions to meet a new sales area that doesn't affect existing customers of the release.
- Error corrections and maintenance.

Frequency:

- Enhancement or Market driven

Audience:

- New customers.
- Existing customers with qualifying contracts
- Existing customers desiring to upgrade to the new feature set

Revision Release

- A Revision (Rev) is a build of all or part of the software that is initially distributed to an internal audience, specifically software quality assurance, for software validation.
- If the Rev is successfully validated and accepted, this version is “released” to manufacturing. If defects are found that prevent the Rev from successfully being validated and prevent the release to manufacturing, the Rev value will be incremented prior to the next validation cycle.

Scope:

- Releasing build to QA team for validation

Frequency:

- As necessary, but typically every 1 to 3 months.

Audience:

- Existing Customers

Build Release

- A Build Release is a build of all or part of the software distributed to an internal audience, these releases should have targeted feature enhancements and issue resolution documented to allow testing in the targeted/specific areas where the changes were implemented.

Scope:

- Resolves a particular defect, typically of a critical Severity level with no viable workaround.

Frequency:

- Extensively used during internal development.

Audience:

- Internal developers
- Internal verification

Verification

Verification is a set of activities that ensures correct implementation of specific functions in a software.

Why Verification:

- If verification is not performed at early stages, there are always a chance of mismatch between the required product and the delivered product.
- Verification exposes more errors.
- Early verification decreases the cost of fixing bugs.
- Early verification enhances the quality of software.

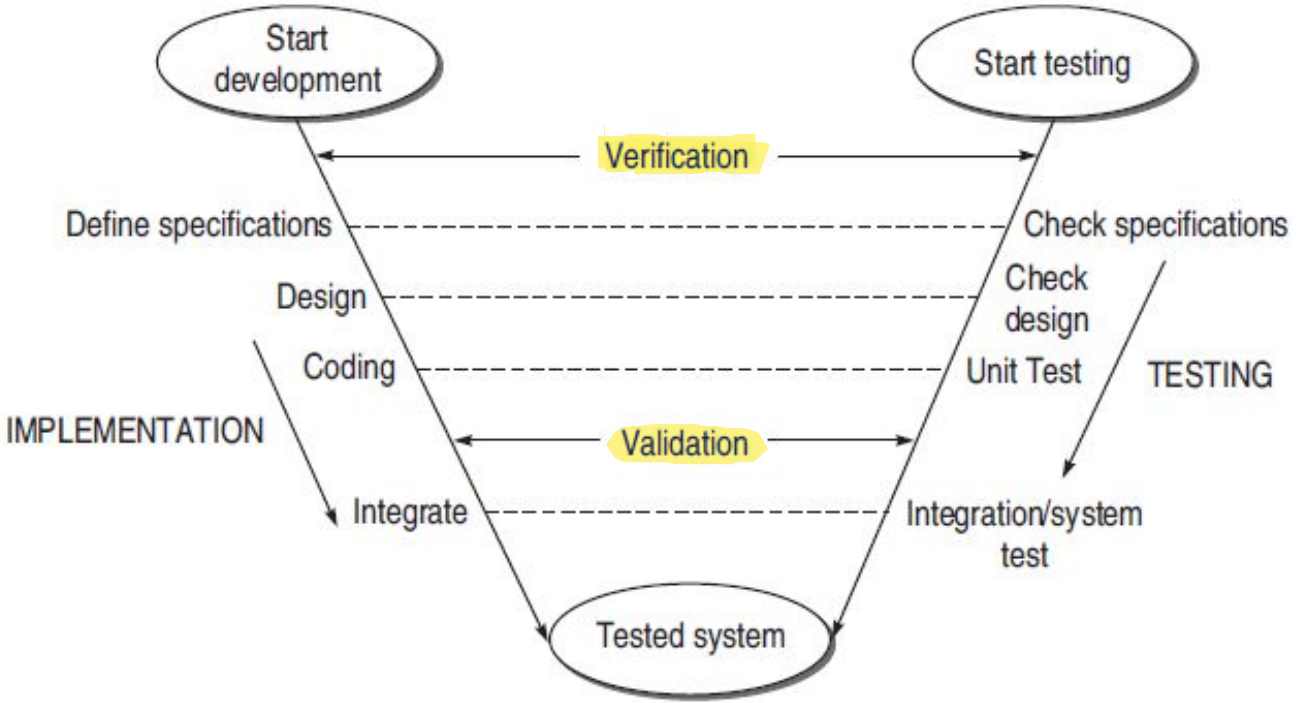
Validation

Validation is a set of activities that ensures the software under consideration has been built right and is traceable to customer requirements.

Why Validation:

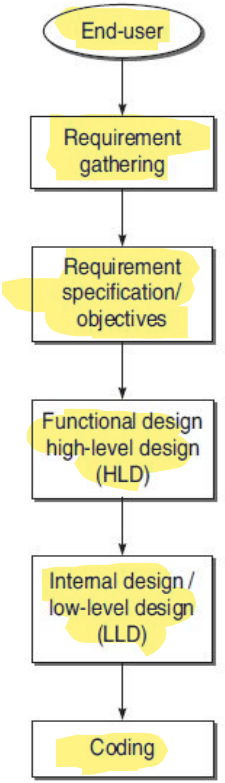
- ✔ To determine whether the product satisfies the users' requirements, as stated in the requirement specification.
- ✔ To determine whether the product's actual behavior matches the desired behavior, as described in the functional design specification.
- ✔ Validation testing provides the last chance to discover bugs, otherwise these bugs will move to the final product released to the customer.
- ✔ Validation enhances the quality of software.

The V Shape Model



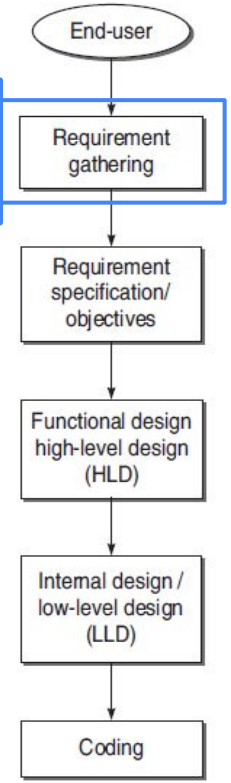
Verification & Validation Activities

Activities in different SDLC phases



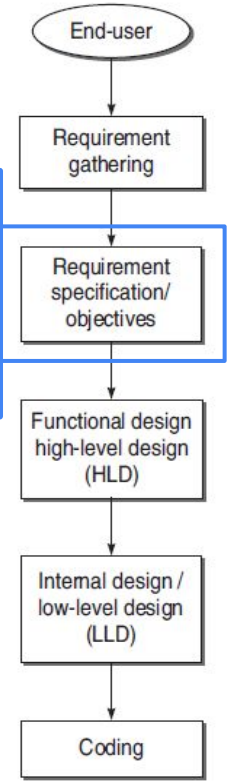
Verification & Validation Activities

The needs of the user are gathered and translated into a written set of requirements.



Verification & Validation Activities

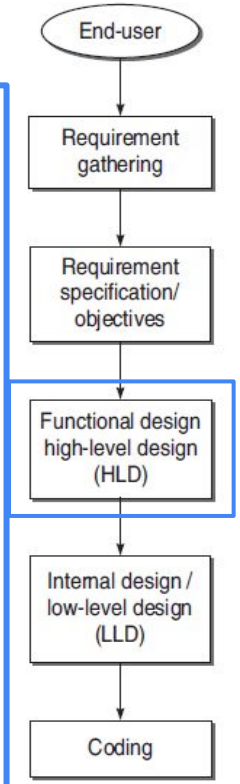
In this phase, all the user requirements are specified in developer's terminology. The specified objectives from the full system which is going to be developed, are prepared in the form of a document known as software requirement specification (SRS).



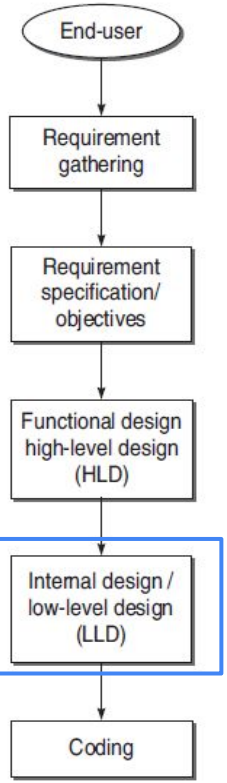
Verification & Validation Activities

The high-level design is prepared with SRS and software analysts convert the requirements into a
usable product. HLD document will contain the following items at a macro level:

1. Overall architecture diagrams along with technology details
2. Functionalities of the overall system with the set of external interfaces
3. List of modules
4. Brief functionality of each module
5. Interface relationship among modules including dependencies between modules, database tables identified along with key elements

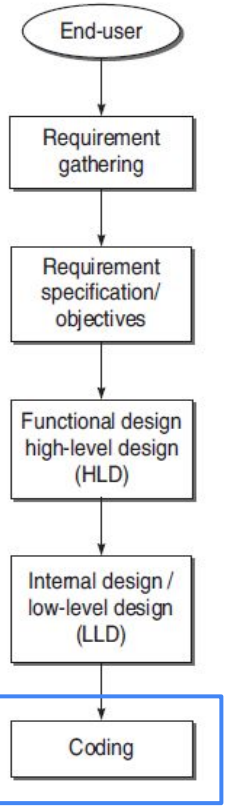


Verification & Validation Activities



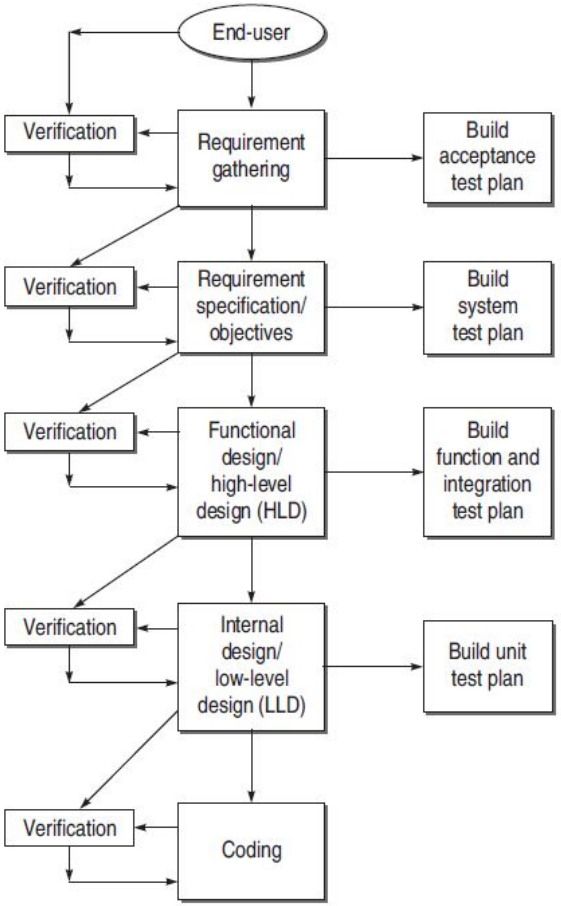
The analysts prepare a micro-level design document called internal design or low-level design (LLD). This document describes each and every module in an elaborate manner, so that the programmer can directly code the program based on this.

Verification & Validation Activities



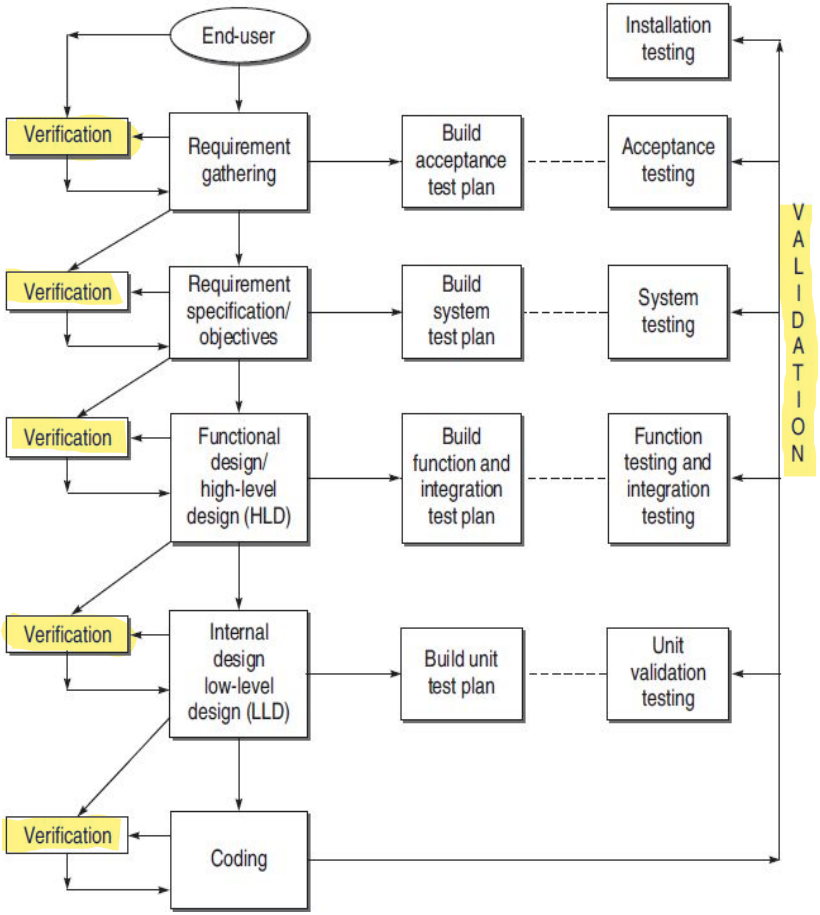
If an LLD document is prepared for every module, then it is easy to code the module. Thus in this phase, using design document for a module, its coding is done.

Verification & Validation Activities



- Tester performs verification and prepares test plan during every SDLC phase
- Test plans are used in validation activities performed after coding the system.

Verification & Validation Activities



Verification activities:

- Verification of Requirements and Objectives
- Verification of High-Level Design
- Verification of Low-Level Design
- Verification of Coding (Unit Verification)

Validation activities are divided into:

- Validation test plan
- Validation test execution

Verification Activities: Requirements

Verification of Requirements

Two parallel activities are performed by the tester:

- The tester reviews the Acceptance criteria in terms of its completeness, clarity, and testability.
- The tester prepares the *Acceptance Test Plan* which is referred at the time of *Acceptance Testing.*

Verification of Objectives/Specifications

Two parallel activities are performed by the tester:

- Verifies all the objectives mentioned in SRS. The purpose of this verification is to ensure that the user's needs are properly understood before proceeding with the project.
- The tester also prepares the *System Test Plan* which is based on SRS. This plan will be referenced at the time of *System Testing*.

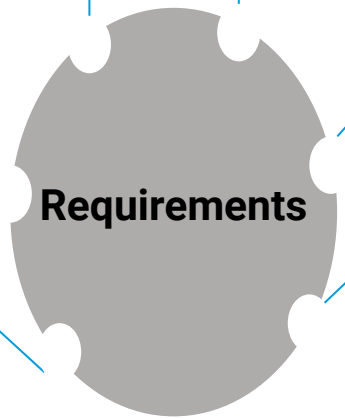
How to verify Requirements & Objectives

Every requirement specified in the SRS must be verified.
Following are the points against which every requirement should be verified:

Correctness
- Each requirement accurately represents some desired feature in the final system

Completeness
- All desired features or characteristics are specified
- Completeness and correctness strongly related

Unambiguous
- Each req has exactly one meaning
- Without this errors will creep in
- Important as natural languages often used



Consistent
- two requirements don't contradict each other

Verifiability
- There must exist a cost effective way of checking if sw satisfies requirements

Traceable
- The origin of the req, and how the req relates to software elements can be determined

Verification Activities: HLD

High-level design takes the second place in SDLC, wherein there is a high probability of finding bugs.
The tester is responsible for two parallel activities in this phase as well:

- The tester verifies the high-level design. all the interfaces and interactions of user/customer (or any person who is interfacing with system) are specified in this phase. The tester verifies that all the components and their interfaces are in tune with requirements of the user. Every requirement in SRS should map the design.
- The tester also prepares a *Function Test Plan* which is based on the SRS. This plan will be referenced at the time of *Function Testing*.

How to verify HLD

- 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 consistency of databases and data warehouses with the requirements specified in SRS.

How to verify HLD

- Check that every functional requirement in the SRS has been take care of in this design.
- Check whether all exceptions handling conditions have been taken care of.
- Verify the process of transform mapping and transaction mapping, used for the transition from requirement model to architectural design.
- Since architectural design deals with the classification of a system into subsystems or modules, check the functionality of each module according to the requirements specified.
- Check the inter-dependence and interface between the modules.

How to verify HLD

- Check all the interfaces between modules according to the 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 the response time for all the interfaces are within required ranges.
- It is very essential for the projects related to real-time systems where response time is very crucial.
- For a Help Facility, verify the following:

  (i) The representation of Help in its desired manner

  (ii) The user returns to the normal interaction from Help
- For error messages and warnings, verify the following:

  (i) Whether the message clarifies the problem

  (ii) Whether the message provides constructive advice for recovering from the error
- For typed command interaction, check the mapping between every menu option and their corresponding commands.

Verification Activities: LLD

In LLD, a detailed design of modules and data are prepared such that an operational software is ready. Every operational detail of each module is prepared.
Testers perform the following parallel activities in this phase:

- The tester verifies the LLD. The details and logic of each module is verified such that the high-level and low-level abstractions are consistent.
- The tester also prepares the Unit Test Plan which will be referred at the time of Unit Testing

How to verify LLD

Some points to be considered to verify LLD:

- Verify the SRS of each module.
- Verify the SDD of each module.
- In LLD, data structures, interfaces, and algorithms are represented by design notations; 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 design, each requirement mentioned in the SRS is verified.

Sno	Req ID	Req Desc	TC ID	TC Desc	Test Design	Test Designer	UAT Test Req?	Test Execution			Defects?	Defect ID	Defect Status	Req Coverage Status
								Test Env	UAT Env	Prod Env				
1	Req01	Login to the Application	TC01	Login with Invalid Username and valid password	Completed	XYZ	No	Passed	No Run	No Run	None	None	N/A	Partial
2			TC02	Login with Valid Username and invalid password	Completed	YZA	No	Passed	No Run	No Run	None	None	N/A	Partial
3			TC03	Login with valid credentials	Completed	XYZ	Yes	Passed	Passed	No Run	Yes	DFCT001	Test OK	Partial

Sheet1 Sheet2 Sheet3

Verification Activities: Code

Since low-level design is converted into source code using some language, there is a possibility of deviation from the LLD. The points against which the code must be verified are:

- Check that every design specification in HLD and LLD has been coded using traceability matrix.
- Examine the code against a language specification checklist.

How to verify Code

Code verification can be done most efficiently by the developer, as he has prepared the code. Some points against which the code can be verified are:

1. Misunderstood or incorrect arithmetic precedence
2. Mixed mode operations
3. Incorrect initialization
4. Precision inaccuracy
5. Incorrect symbolic representation of an expression
6. Different data types
7. Improper or non-existent loop termination
8. Failure to exit

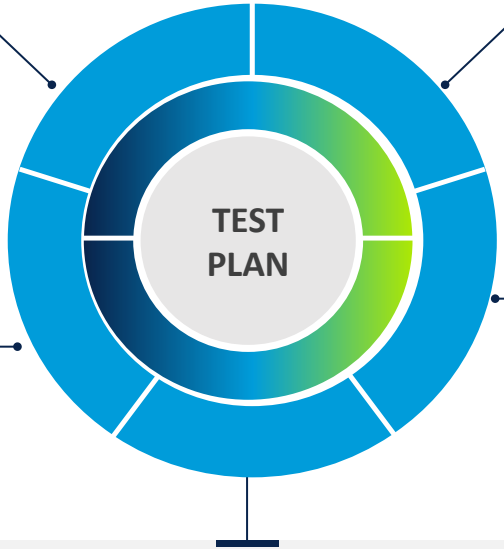
How to verify Code

- As verification of code can not be done for whole system we perform the code verification for modules.
- This is also known as Unit Verification Testing
- Unit verification is largely white-box oriented.
- points to be considered while performing unit verification:
  - 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 also.
  - All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once.
  - All error handling paths are tested.

Validation Activities: Validation Test Plan

For preparing a validation test plan, testers must follow the points described below:

- Testers must understand the current SDLC phase.
- Testers must study the relevant documents in the corresponding SDLC phase.
- On the basis of the understanding of SDLC phase and related documents, testers must prepare the related test plans which are used at the time of validation testing. Under test plans, they must prepare a sequence of test cases for validation testing.



- is prepared in the requirement phase according to the acceptance criteria prepared from the user feedback.
- is used at the time of Acceptance Testing.

- is prepared to verify the objectives specified in the SRS.
- is used at the time of System Testing.

- is prepared in the LLD phase.
- is used at the time of Unit Testing.

- is prepared in the HLD phase.
- is used at the time of Function Testing.

- is prepared to validate the integration of all the modules such that all their interdependencies are checked.
- is used at the time of Integration Testing.

Validation Activities: Validation Test Execution

Validation test execution can be divided in the following testing activities:

- Unit Validation testing
- Integration testing
- Function testing
- System Testing
- Acceptance testing
- Installation testing