Noakhali Science and Technology University

Institute of Information Technology

Software Engineering

Course Title: Software Testing and Quality Assurance

Course Code: SE3209

Assignment on: Verification and Validation (3rd Chapter)

Submitted to
Tasniya Ahmed
Lecturer of IIT, NSTU

Submitted by
MD Mynuddin
Roll: ASH1825007M
1st Batch
IIT, NSTU

Date: 5 April 2021

⇒ "Varification in Software testing" in a process of checking documents, design, code and program in order to check if the Software has been built according to the requirement or not. The main goal of varification process is to ensure quality of Software application, design, architecture etc. The varification process involves activities like reviews, walk through and inspection.

⇒ "Validation" in Software testing, in a dynamic mechanism of testing and validating if the Software product actually meets the exact needs of the customer or not. The Process helps to ensure that the Software fulfills the desired use in a appropriate environment.

The validation process involves activities like unit testing, integration testing, System testing and user acceptance testing.

⇒ Key Difference

→ Verification process includes checking of documents, design, code and program whereas validation process includes testing and validation of the actual product.

→ Verification does not involve code execution while validation involves code execution.

→ Verification use methods like reviews, walkthroughs, inspection and desk checking. whereas Validation use methods like black box testing, black box testing and non-functional testing.

→ Verification process targets on Software architecture, design, database etc.

where validation process targets the actual software Product.

→ Verification checks whether the software confirms a specification whereas validation checks whether the software meets the requirements and expectations.

→ Verification finds the bugs early in the development cycle whereas validation finds the bugs that verification can not catch.

→ Verification process comes before validation. whereas Validation comes after verification.

⇒ Testing strategy of varification and validation on V- diagram.

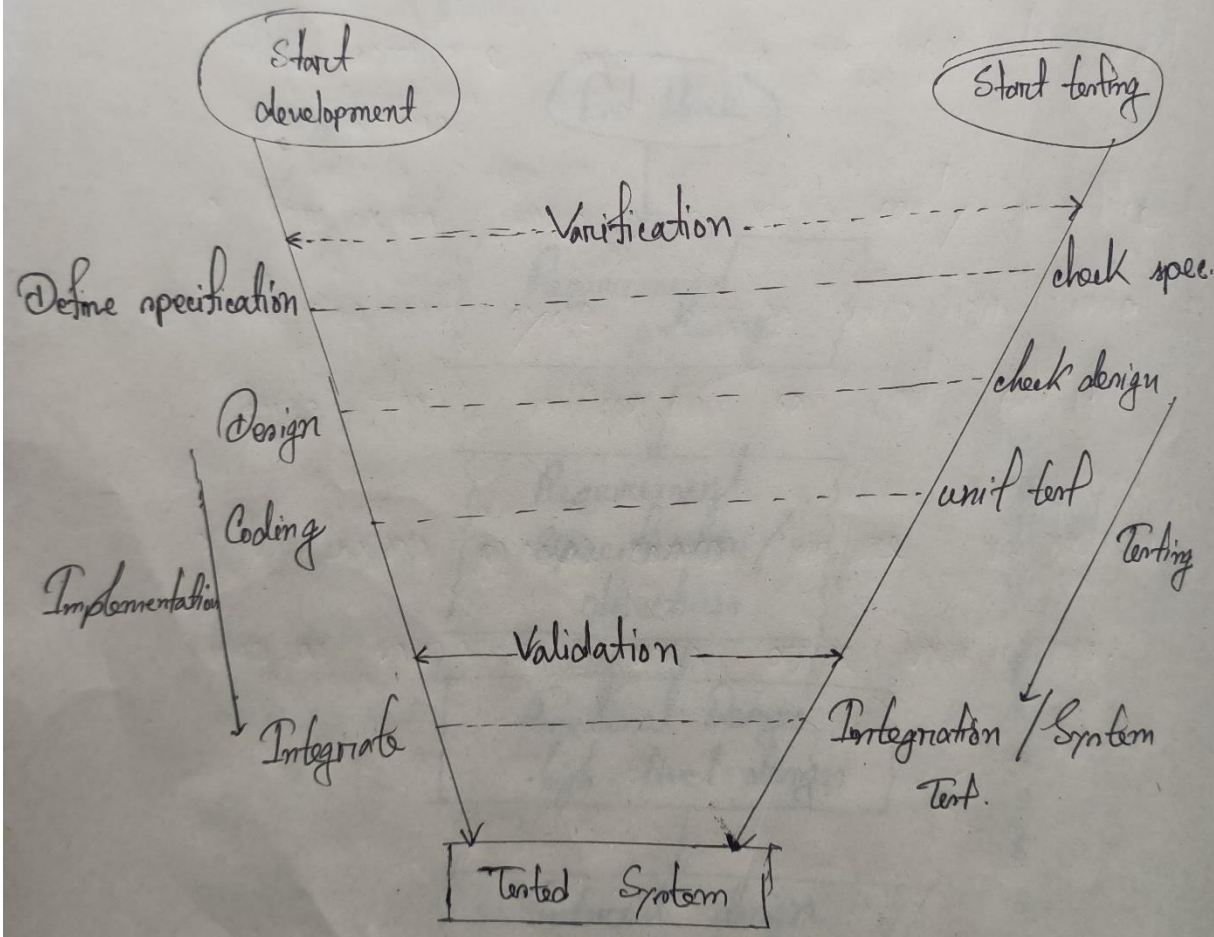
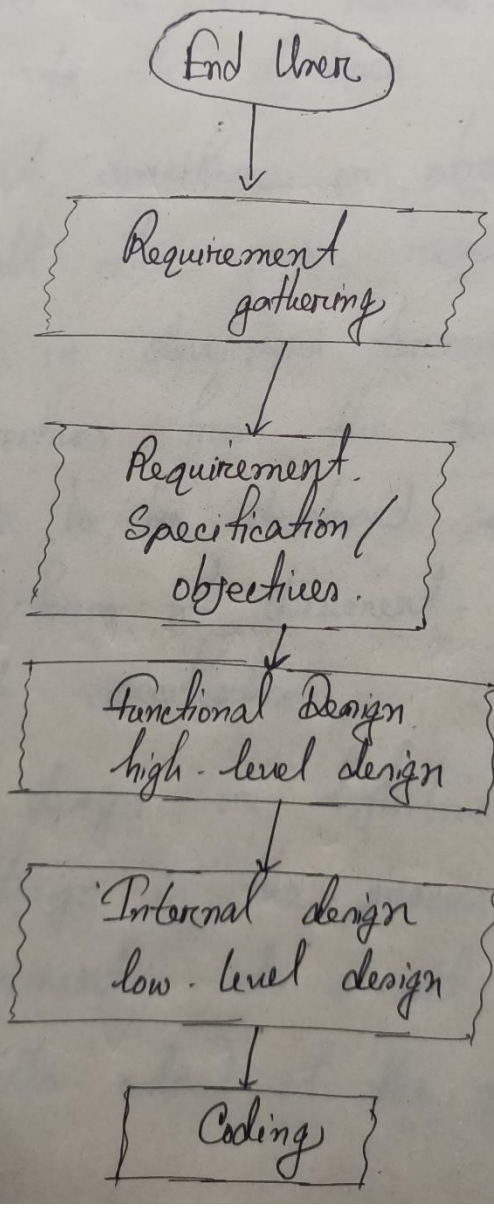


figure : V- testing.

⇒ Verification and Validation (V & V) Activities.

Let's understand SDLC phases. After this, verification and validation activities in those SDLC phases will be described.



⇒"Requirement gathering": The needs of the users are gathered and translated into a written set of requirements. These requirements are prepared from the users viewpoint only and do not include any technicalities according to the developer.

⇒"Requirement Specification or objectives": In this phase, all the user requirements are specified in developers terminology. The specified objectives from the full System which is going to be developed are prepare in the form of document. known as Software requirement specification.

⇒"functional design or high-level design": functional design is the process of translating users requirements into a set of external Interfaces. The output of the process is the

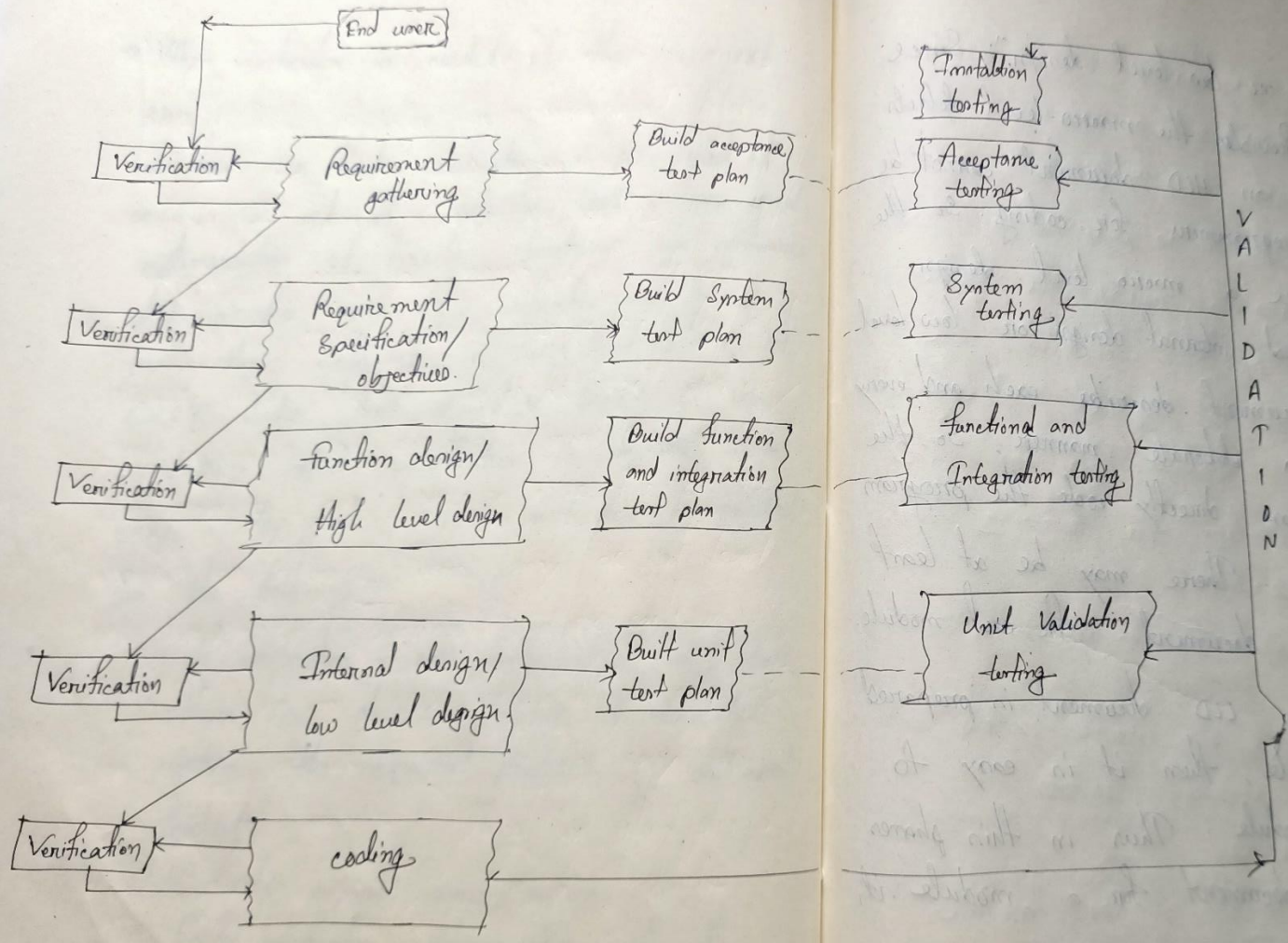
functional design specification, which describes the product behaviour as seen by an observer external to the product.

The high-level design is prepared with SRS and Software analysts convert the requirement into a usable product. In HLD, the Software System architecture is prepared and broken into independent modules. Thus, and 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.

⇒ "Internal design or low level design": Since HLD Product Provides the macro-level details of a System, an HLD document cannot be given to programmers for coding. So the analysts prepare a macro-level design document called internal design or low level design. This document describe each and every module in a elaborate manner. So, the Programmer can directly code the program based on this. There may be at least one separate document for each module.

⇒ "Coding": In 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.



when the coding is over for a unit or a System and parallel Verification activities can be performed now, then the System can be validate. It means "validation" activities can be performed now.

figure: Never Perform verification and prepare test plan during every SDLC phases.

⇒ Why verification is needed of the steps of SDLC.

we have seen that verification is performed at or in a set of activities that ensure correct implementation of specific functions in a software. what is the need of verification? Can't we test the Software in the final phases of SDLC? Under the V and V Process, it is mandatory that verification is performed at every step of SDLC. Verification is needed for the following:

→ If verification is not performed at early stages there are always a chance of mismatch between the required product and delivered Product.

→ Verification expose more errors.

→ Early verification decrees the cost of fixing bugs.

→ Enhance the quality of Software.

⇒ Goals of Verification.

"Everything must be verified"

In Principle, all the SDL phases and all the products of these process must be verified.

"Result of verification may not be Binary"

Verification may not just be the acceptance or rejection of a product. There are many verification activities whose result cannot be reduce to a binary answer. Often one has to accept approximations. for example, sometimes correctness of a requirement can't be rejected or acceptance outright, but can be accepted with a degree of satisfaction or rejected with a certain degree of modification.

"Even Implicit Qualities must be verified"

The qualities desired in the Software are explicitly stated in the SRS. But those requirements which are implicit and not mentioned anywhere must also be Verified.

⇒ Verification Activities.

All the verification activities are performed in connection with the different phases SDLC. The following verification activities have been identified.

- → Verification of Requirements and Objectives
- → Verification of HLD
- → Verification of LLD
- → Verification of coding.

⇒ what are the points against which verification of requirements will be done?

OR.

How to verify Requirements and objectives?

→ following are the points against which verification of requirement in SRS should be verified.

⇒ Correctness: There are no tools or procedure to measure the correctness of a specification. The tester uses his or her intelligence to verify the correctness of requirement. following some points which can be adopted.

i. Tester should refer to other documentation or applicable standards and compare the specified requirement with them.

ii. Tester can interact with customer or users if requirements are not well understood.

iii. Testers should check the correctness in the sense of realistic requirement.

If the tester feels that a requirement cannot be realized with existing hardware and Software technology. It means that on in that case, the requirement should either be updated or remove from SRS.

- Consistant: No specification should contradict or conflict with another. Conflicts produce bugs in the next stages, therefor the must be checked for the followings:

1. Real world object conflict. For example one specification recomends mouse for input, another recomends joystick.
2. Logical Conflict.
3. Conflicts in terminology should also verified. For example, at one place the tester term process is used while at another place, it has been termed as task or module.

⇒ Completeness: The requirements specified in the SRS must be verified for Completeness.

we must,

i. Verify that all significant requirement such as functionality, performance, design constraints, attribute or external interfaces are complete.

ii. check weather response of every possible input (valid, invalid) to the software have been defined.

⇒ Updation: Requirement specifications are not stable they may be modified or another requirement may be added later. Therefore if any updation in there in the SRS then the update spec. must be verified.

1. If a specification in new one than all the above mention step's and their feasibility should be verified.

11. If the specification is a change in an already mention specification - than we must verify them and this change can be implemented in the current design.

⊙ Traceability : The traceability of requirement must also be verified such that the origin of each req. is clear and also weather it facilities referencing in future development or enhancement - - documentation. The following two types of traceability must be verified.

1. "Backward traceability" check that each requirement referrences its sources in previous document.

11. "forward traceability": check that each req. has a unique name or reference number in all the documents. forward traceability assumes more meaning than this, but for the sake of clearity. here it should be understood in the sense that every requirement has been recognized in other document.

= How to verify high-level design?

If a bug goes undetected in the high-level-design phase. Therefore verification for high level design must be done very carefully. This design is divided into three parts.

1. Data Design.
11. Architecture design.
111. Interface design.

→ How to verify low level design?

For verification, the SRS and SDD in individual modules are referred to. Some points to be considered are listed below.

i. Verify the SRS of each module.

ii. Verify the SDD of each module.

iii. In LLD, data structures, interfaces, and algorithms are represented by design notation. verify the consistency of every item with their design notation.

→ How to verify code?

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

i. Check the every design Specification in HLD and LLD has been coded using traceability matrix.

ii. Examine the code against a language spec checklist.

iii. Code Verification can be done most efficiently by the developer, as he has prepared the code. He can verify every statement, Control structure, loop and logic such that every possible method of application execution is tested. In this way, he verifies the whole module which he has developed.

Some point's against which the code can be verified are:

→ Misunderstood or incorrect arithmatic Precedence

→ Mixed Module operation.

→ Incorrect initialization.

→ Precision inaccuracy.

→ Incorrect Symbolic representation of an expression.

→ Different data types.

→ Improper or non-existent loop termination.

→ failure to exit.

iv. Dynamic testing

⇒ Unit Verification. Verification of coding cannot be done for the whole System. Moreover, the system is divided into modules. Therefore, verification of coding means the verification of code of modules by their developers. This is also known as unit verification testing. Listed below are the points to be considered while Performing unit Verification:

i. Interfaces are verified to ensure that information properly flows in and out of the program unit under test.

ii. The local data structure is verified to maintain data integrity.

iii. Boundary conditions are checked to verify that the module is working fine on boundaries also.

iv. All independent paths through the control structure are exercised to ensure that all statements in a module have been execute at least once.

v. All error handaling paths are tested.

Unit verification is largely white-box oriented.