

What is Software Testing?

Software testing is a process of executing a program or application with the intent of finding the **software bugs**.

- It can also be stated as the **process of validating and verifying** that a software program or application or product:
 - Meets the business and technical requirements that guided its design and development
 - Works as expected
 - Can be implemented with the same characteristic.

Let's break the definition of **Software testing** into the following parts:

1) Process: Testing is a process rather than a single activity.

2) All Life Cycle Activities: Testing is a process that's take place throughout the **Software Development Life Cycle (SDLC)**.

- The process of designing tests early in the life cycle can help to prevent defects from being introduced in the code. Sometimes it's referred as "**verifying the test basis via the test design**".
- The **test basis** includes documents such as the requirements and design specifications.

3) Static Testing: It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing. For example: reviewing, walkthrough, inspection, etc.

4) Dynamic Testing: In dynamic testing the software code is executed to demonstrate the result of running tests. It's done during validation process. For example: unit testing, integration testing, system testing, etc.

5) Planning: We need to plan as what we want to do. We control the test activities, we report on testing progress and the status of the software under test.

6) Preparation: We need to choose what testing we will do, by selecting test conditions and **designing test cases**.

7) Evaluation: During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.

8) Software products and related work products: Along with the testing of code the testing of requirement and design specifications and also the related documents like operation, user and training material is equally important.

Why is software testing necessary?

Software Testing is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous. We need to check everything and anything we produce because things can always go wrong – **humans make mistakes all the time**.

Since we assume that our work may have mistakes, hence we all need to check our own work. However some mistakes come from bad assumptions and blind spots, so we might make the same mistakes when we check our own work as we made when we did it. So we may not notice the flaws in what we have done.

Ideally, we should get someone else to check our work because another person is more likely to spot the flaws.

There are several reasons which clearly tell us as why Software Testing is important and what are the major things that we should consider while testing of any product or application.

Software testing is very important because of the following reasons:

1. Software testing is really required to point out the defects and errors that were made during the development phases.
2. It's essential since it makes sure of the Customer's reliability and their satisfaction in the application.
3. It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence.
4. Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.
5. Testing is required for an effective performance of software application or product.
6. It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.
7. It's required to stay in the business.

What are software testing objectives and purpose?

Software Testing has different goals and objectives. The major objectives of Software testing are as follows:

- **Finding defects** which may get created by the programmer while developing the software.
- Gaining confidence in and providing information about the level of quality.
- To prevent defects.
- To make sure that the end result meets the business and user requirements.
- To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
- To gain the confidence of the customers by providing them a quality product.

Software testing helps in finalizing the software application or product against business and user requirements. It is very important to have good test coverage

in order to test the software application completely and make it sure that it's performing well and as per the specifications.

While determining the coverage the test cases should be designed well with maximum possibilities of finding the errors or bugs. The test cases should be very effective. This objective can be measured by the number of defects reported per test cases. Higher the number of the defects reported the more effective are the test cases.

Once the delivery is made to the end users or the customers they should be able to operate it without any complaints. In order to make this happen the tester should know as how the customers are going to use this product and accordingly they should write down the test scenarios and design the test cases. This will help a lot in fulfilling all the customer's requirements.

Software testing makes sure that the testing is being done properly and hence the system is ready for use. Good coverage means that the testing has been done to cover the various areas like functionality of the application, compatibility of the application with the OS, hardware and different types of browsers, performance testing to test the performance of the application and load testing to make sure that the system is reliable and should not crash or there should not be any blocking issues. It also determines that the application can be deployed easily to the machine and without any resistance. Hence the application is easy to install, learn and use.

What is Defect or bugs or faults in software testing?

Definition:

- A defect is an error or a bug, in the application which is created. A programmer while designing and building the software can make mistakes or error. These mistakes or errors mean that there are flaws in the software. These are called defects.

- When actual result deviates from the expected result while testing a software application or product then it results into a defect. Hence, any deviation from the specification mentioned in the product functional specification document is a defect. In different organizations it's called differently like bug, issue, incidents or problem.

- When the result of the software application or product does not meet with the end user expectations or the software requirements then it results into a Bug or Defect. These defects or bugs occur because of an error in logic or in coding which results into the failure or unpredicted or unanticipated results.

Additional Information about Defects / Bugs:

While testing a software application or product if large number of defects are found then it's called Buggy.

When a tester finds a bug or defect it's required to convey the same to the developers. Thus they report bugs with the detail steps and are called as Bug Reports, issue report, problem report, etc.

This Defect report or Bug report consists of the following information:

- **Defect_ID** – Every bug or defect has it's unique identification number

- **Defect Description** – This includes the abstract of the issue.
- **Product Version** – This includes the product version of the application in which the defect is found.
- **Detail Steps** – This includes the detailed steps of the issue with the screenshots attached so that developers can recreate it.
- **Date Raised** – This includes the Date when the bug is reported
- **Reported By** – This includes the details of the tester who reported the bug like Name and ID
- **Status** – This field includes the Status of the defect like New, Assigned, Open, Retest, Verification, Closed, Failed, Deferred, etc.
- **Fixed by** – This field includes the details of the developer who fixed it like Name and ID
- **Date Closed** – This includes the Date when the bug is closed
- **Severity:** Based on the severity (Critical, Major or Minor) it tells us about impact of the defect or bug in the software application
- **Priority:** Based on the Priority set (High/Medium/Low) the order of fixing the defect can be made.

What is a Failure in software testing?

If under certain environment and situation defects in the application or product get executed then the system will produce the wrong results causing a failure.

Not all defects result in failures, some may stay inactive in the code and we may never notice them. Example: Defects in dead code will never result in failures.

It is not just defects that give rise to failure. Failures can also be caused because of the other reasons also like:

- Because of the environmental conditions as well like a radiation burst, a strong magnetic field, electronic field or pollution could cause faults in hardware or firmware. Those faults might prevent or change the execution of software.

- **Failures may also arise** because of human error in interacting with the software, perhaps a wrong input value being entered or an output being misinterpreted.
- Finally failures may also be caused by someone deliberately trying to cause a failure in the system.

Difference between Error, Defect and Failure in software testing:

Error: The mistakes made by programmer is known as an 'Error'. This could happen because of the following reasons:

- Because of some confusion in understanding the functionality of the software
- Because of some miscalculation of the values
- Because of misinterpretation of any value, etc.

Defect: The bugs introduced by programmer inside the code are known as a defect. This can happen because of some programming mistakes.

Failure: If under certain circumstances these defects get executed by the tester during the testing then it results into the failure which is known as software failure.

Few points that are important to know:

- When tester is executing a test he/she may observe some difference in the behavior of the feature or functionality, but this not because of the failure. This may happen because of the wrong test data entered, tester may not be aware of the feature or functionality or because of the bad environment. Because of these reasons incidents are reported. They are known as incident report. The condition or situation which requires further analysis or clarification is known as incident. To deal with the incidents the programmer need to do the analysis that whether this incident has occurred because of the failure or not.

- It's not necessary that defects or bugs introduced in the product are only by the software. To understand it further let's take an example. A bug or defect can also be introduced by a business analyst. Defects present in the specifications like requirements specification and design specifications can be detected during the reviews. When the defect or bug is caught during the review cannot result into failure because the software has not yet been executed.
- These defects or bugs are reported not to blame the developers or any people but to judge the quality of the product. The quality of product is of utmost importance. To gain the confidence of the customers it's very important to deliver the quality product on time.

From where do defects and failures in software testing arise?

Defects and failures basically arise from:

- **Errors in the specification**, design and implementation of the software and system
- Errors in use of the system
- Environmental conditions
- Intentional damage
- Potential consequences of earlier errors

Errors in the specification and design of the software:

Specification is basically a written document which describes the functional and non – functional aspects of the software by using prose and pictures. For testing specifications there is no need of having code. Without having code we can test the specifications. About 55% of all the bugs present in the product are because of the mistakes present in the specification. Hence testing the specifications can lots of time and the cost in future or in later stages of the product.

Errors in use of the system:

Errors in use of the system or product or application may arise because of the following reasons:

- Inadequate knowledge of the product or the software to the tester. The tester may not be aware of the functionalities of the product and hence while testing the product there might be some defects or failures.
- Lack of the understanding of the functionalities by the developer. It may also happen that the developers may not have understood the functionalities of the product or application properly. Based on their understanding the feature they will develop may not match with the specifications. Hence this may result into the defect or failure.

Environmental conditions:

Because of the wrong setup of the testing environment testers may report the defects or failures. As per the recent surveys it has been observed that about 40% of the tester's time is consumed because of the environment issues and this has a great impact on quality and productivity. Hence proper test environments are required for quality and on time delivery of the product to the customers.

Intentional damage:

The defects and failures reported by the testers while testing the product or the application may arise because of the intentional damage.

Potential consequences of earlier errors:

Errors found in the earlier stages of the development reduce our cost of production. Hence it's very important to find the error at the earlier stage. This could be done by reviewing the specification documents or by walkthrough. The downward flow of the defect will increase the cost of production.

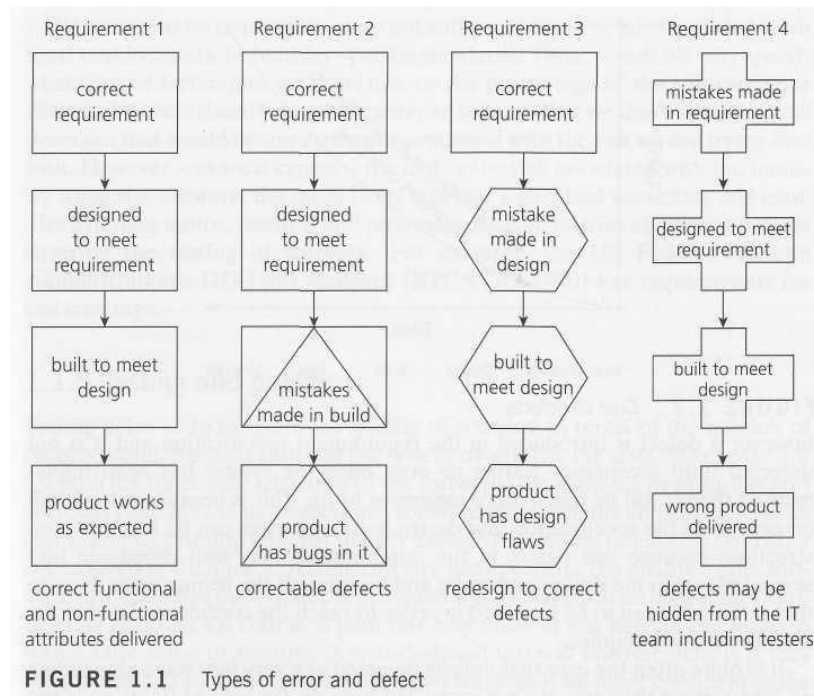
When do defects in software testing arise?

Because of the following reasons the software defects arise:

- The person using the software application or product may not have enough knowledge of the product.
- Maybe the software is used in the wrong way which leads to the defects or failures.
- The developers may have coded incorrectly and there can be defects present in the design.
- Incorrect setup of the testing environments.

To know when defects in software testing arise, let us take a small example with a diagram as given below.

We can see that **Requirement 1** is implemented correctly – we understood the customer's requirement, designed correctly to meet that requirement, built correctly to meet the design, and so deliver that requirement with the right attributes: functionally, it does what it is supposed to do and it also has the right non-functional attributes, so it is fast enough, easy to understand and so on.



With the other requirements, errors have been made at different stages.

Requirement 2 is fine until the software is coded, when we make some mistakes and introduce defects. Probably, these are easily spotted and corrected during testing, because we can see the product does not meet its design specification.

The defects introduced in **Requirement 3** are harder to deal with; we built exactly what we were told to but unfortunately the designer made some mistakes so there are defects in the design. Unless we check against the requirements definition, we will not spot those defects during testing. When we do notice them they will be hard to fix because design changes will be required.

The defects in **Requirement 4** were introduced during the definition of the requirements; the product has been designed and built to meet that flawed requirements definition. If we test the product meets its requirements and design, it will pass its tests but may be rejected by the user or customer. Defects reported by the customer in acceptance test or live use can be very costly. Unfortunately, requirements and design defects are not rare; assessments of thousands of projects have shown that defects introduced during requirements and design make up close to half of the total number of defects.

What is the cost of defects in software testing?

The cost of defects can be measured by the impact of the defects and when we find them. Earlier the defect is found lesser is the cost of defect. For example if error is found in the requirement specifications then it is somewhat cheap to fix it. The correction to the requirement specification can be done and then it can be re-issued. In the same way when defect or error is found in the design then the design can be corrected and it can be re-issued. But if the error is not caught in the specifications and is not found till the user acceptance then the cost to fix those errors or defects will be way too expensive.

If the error is made and the consequent defect is detected in the **requirements phase** then it is relatively cheap to fix it.

Similarly if an error is made and the consequent defect is found in the **design phase** then the design can be corrected and reissued with relatively little expense.

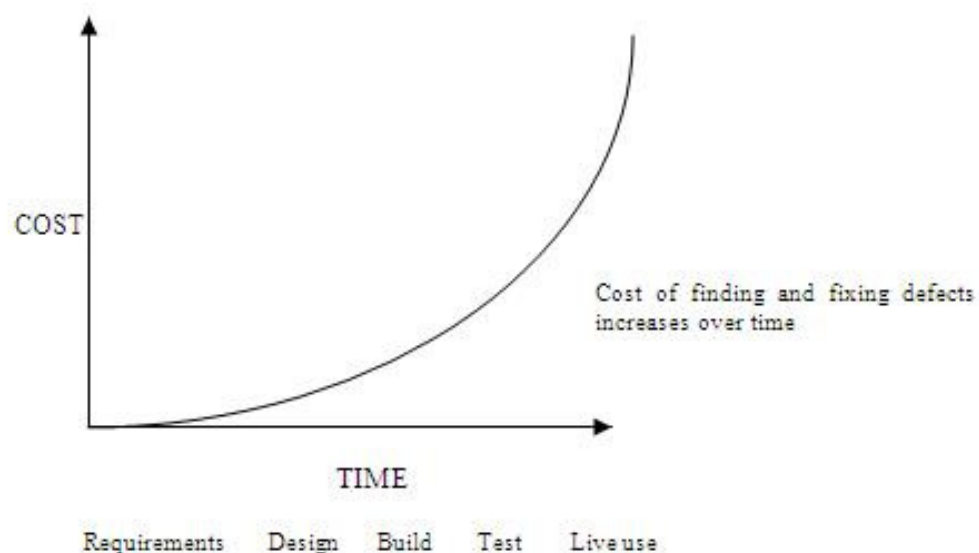


FIGURE 1.2

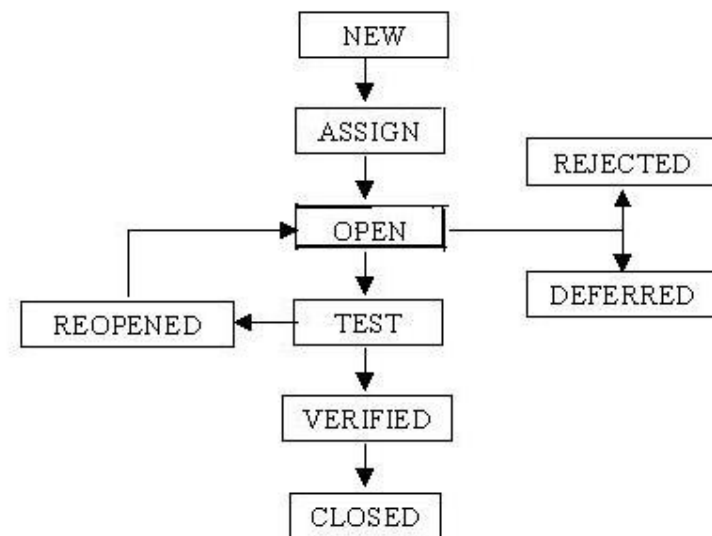
The same applies for **construction phase**. If however, a defect is introduced in the requirement specification and it is not detected until acceptance testing or even once the system has been implemented then it will be much more expensive to fix. This is because rework will be needed in the specification and design before changes can be made in construction; because one defect in the requirements may well propagate into several places in the design and code; and because all the testing work done-to that point will need to be repeated in order to reach the confidence level in the software that we require.

It is quite often the case that defects detected at a very late stage, depending on how serious they are, are not corrected because the cost of doing so is too expensive.

What is a Defect Life Cycle or a Bug lifecycle in software testing?

Defect life cycle is a cycle which a defect goes through during its lifetime. It starts when defect is found and ends when a defect is closed, after ensuring it's not reproduced. **Defect life cycle** is related to the bug found during testing.

The bug has different states in the Life Cycle. The Life cycle of the bug can be shown diagrammatically as follows:



Bug or defect life cycle includes following steps or status:

1. **New:** When a defect is logged and posted for the first time. It's state is given as new.
2. **Assigned:** After the tester has posted the bug, the lead of the tester approves that the bug is genuine and he assigns the bug to corresponding developer and the developer team. It's state given as assigned.

3. **Open:** At this state the developer has started analyzing and working on the defect fix.
4. **Fixed:** When developer makes necessary code changes and verifies the changes then he/she can make bug status as 'Fixed' and the bug is passed to testing team.
5. **Pending retest:** After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.
6. **Retest:** At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.
7. **Verified:** The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to "verified".
8. **Reopen:** If the bug still exists even after the bug is fixed by the developer, the tester changes the status to "reopened". The bug goes through the life cycle once again.
9. **Closed:** Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to "closed". This state means that the bug is fixed, tested and approved.
10. **Duplicate:** If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to "duplicate".
11. **Rejected:** If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to "rejected".
12. **Deferred:** The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are priority of the bug may be low, lack of time for the release or the bug may not have major effect on the software.
13. **Not a bug:** The state given as "Not a bug" if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and field of the application like change of colour of some text then it is not a bug but just some change in the looks of the application.

What is the difference between Severity and Priority?

Here are two key things in defects of the **software testing**. They are:

- 1) Severity
- 2) Priority

What is the difference between Severity and Priority?

1) Severity:

It is the extent to which the **defect** can affect the software. In other words it defines the impact that a given defect has on the system. **For example:** If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.

Severity can be of following types:

- **Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
- **Major:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
- **Moderate:** The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.

- **Minor:** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
- **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

2) Priority:

Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait? This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements. **For example:** If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

Priority can be of following types:

- **Low:** The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect have been fixed.
- **Medium:** The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
- **High:** The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.

Few very important scenarios related to the severity and priority which are asked during the interview:

High Priority & High Severity: An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. A site maintaining the student details, on saving record if it, doesn't allow to save the record then this is high priority and high severity bug.)

High Priority & Low Severity: The spelling mistakes that happens on the cover page or heading or title of an application.

High Severity & Low Priority: An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.

Low Priority and Low Severity: Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

What is Software Quality?

Quality software is reasonably [bug or defect](#) free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable.

ISO 8402-1986 standard defines quality as “the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs.”

Key aspects of quality for the customer include:

- Good design – looks and style
- Good functionality – it does the job well
- Reliable – acceptable level of breakdowns or failure
- Consistency
- Durable – lasts as long as it should
- Good after sales service
- Value for money

Good design – looks and style:

It is very important to have a good design. The application or product should meet all the requirement specifications and at the same time it should be user friendly. The customers are basically attracted by the good looks and style of the application. The right color combinations, font size and the styling of the texts and buttons are very important.

Good functionality – it does the job well:

Along with the good looks of the application or the product it's very important that the functionality should be intact. All the features and their functionality should work as expected. There should not be any deviation in the actual result and the expected result.

Reliable – acceptable level of breakdowns or failure:

After we have tested for all the features and their functionalities it also very important that the application or product should be reliable. For example: There is an application of saving the students records. This application should save all the students records and should not fail after entering 100 records. This is called reliability.

Consistency:

The software should have consistency across the application or product. Single software can be multi dimensional. It is very important that all the different dimensions should behave in a consistent manner.

Durable – lasts as long as it should:

The software should be durable. For example if software is being used for a year and the number of data has exceed 5000 records then it should fail if number of records increases. The software product or application should continue to behave in the same way without any functional breaks.

Good after sales service:

Once the product is shipped to the customers then maintenance comes into the picture. It is very important to provide good sales services to keep the customers happy and satisfied. For example if after using the product for six months the customer realizes to make some changes to the application then those changes should be done as fast as possible and should be delivered to the customers on time with quality.

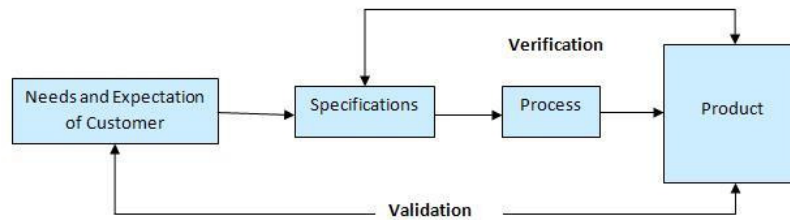
Value for money:

It's always important to deliver the product to the customers which have value for money. The product should meet the requirement specifications. It should work as expected, should be user friendly. We should provide good services to the customers. Other than the features mentioned in the requirement specifications some additional functionality could be given to the customers which they might not have thought of. These additional functionalities should make their product more user friendly and easy to use. This also adds value for money.

What is Verification in software testing? or What is software verification?

- It makes sure that the product is designed to deliver all functionality to the customer.
- Verification is done at the starting of the development process. It includes reviews and meetings, walkthroughs, inspection, etc. to evaluate documents, plans, code, requirements and specifications.
- Suppose you are building a table. Here the verification is about checking all the parts of the table, whether all the four legs are of correct size or not. If one leg of table is not of the right size it will imbalance the end product. Similar behavior is also noticed in case of the software product or application. If any feature of software product or application is not up to the mark or if any defect is found then it will result into the failure of the end

product. Hence, verification is very important. It takes place at the starting of the development process.



Software verification and validation

- It answers the questions like: **Am I building the product right?**
- Am I accessing the data right (in the right place; in the right way).
- It is a Low level activity
- Performed during development on key artifacts, like walkthroughs, reviews and inspections, mentor feedback, training, checklists and standards.
- Demonstration of consistency, completeness, and correctness of the software at each stage and between each stage of the development life cycle.

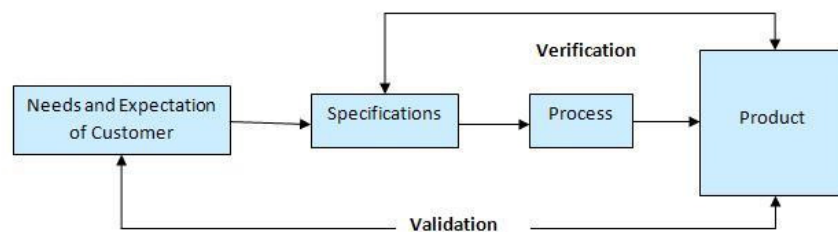
According to the Capability Maturity Model (CMMI-SW v1.1) we can also define verification as the process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Advantages of Software Verification :

1. Verification helps in lowering down the count of the defect in the later stages of development.
2. Verifying the product at the starting phase of the development will help in understanding the product in a better way.
3. It reduces the chances of failures in the software application or product.
4. It helps in building the product as per the customer specifications and needs.

What is Validation in software testing? or What is software validation?

- Determining if the system complies with the requirements and performs functions for which it is intended and meets the organization's goals and user needs.
- Validation is done at the end of the development process and takes place after verifications are completed.
- It answers the question like: **Am I building the right product?**
- Am I accessing the right data (in terms of the data required to satisfy the requirement).
- It is a High level activity.
- Performed after a work product is produced against established criteria ensuring that the product integrates correctly into the environment.
- Determination of correctness of the final software product by a development project with respect to the user needs and requirements.



Software verification and validation

According to the Capability Maturity Model (CMMI-SW v1.1) we can also define validation as The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

A product can pass while verification, as it is done on the paper and no running or functional application is required. But, when same points which were verified on the paper is actually developed then the running application or product can fail while validation. This may happen because when a product or application is build as per the specification but these specifications are not up to the mark hence they fail to address the user requirements.

Advantages of Validation:

During verification if some defects are missed then during validation process it can be caught as failures.

1. If during verification some specification is misunderstood and development had happened then during validation process while executing that functionality the difference between the actual result and expected result can be understood.
2. Validation is done during testing like feature testing, integration testing, system testing, load testing, compatibility testing, stress testing, etc.
3. Validation helps in building the right product as per the customer's requirement and helps in satisfying their needs.

Validation is basically done by the testers during the testing. While validating the product if some deviation is found in the actual result from the expected result then a bug is reported or an incident is raised. Not all incidents are bugs. But all bugs are incidents. Incidents can also be of type 'Question' where the functionality is not clear to the tester.

Hence, validation helps in unfolding the exact functionality of the features and helps the testers to understand the product in much better way. It helps in making the product more user friendly.