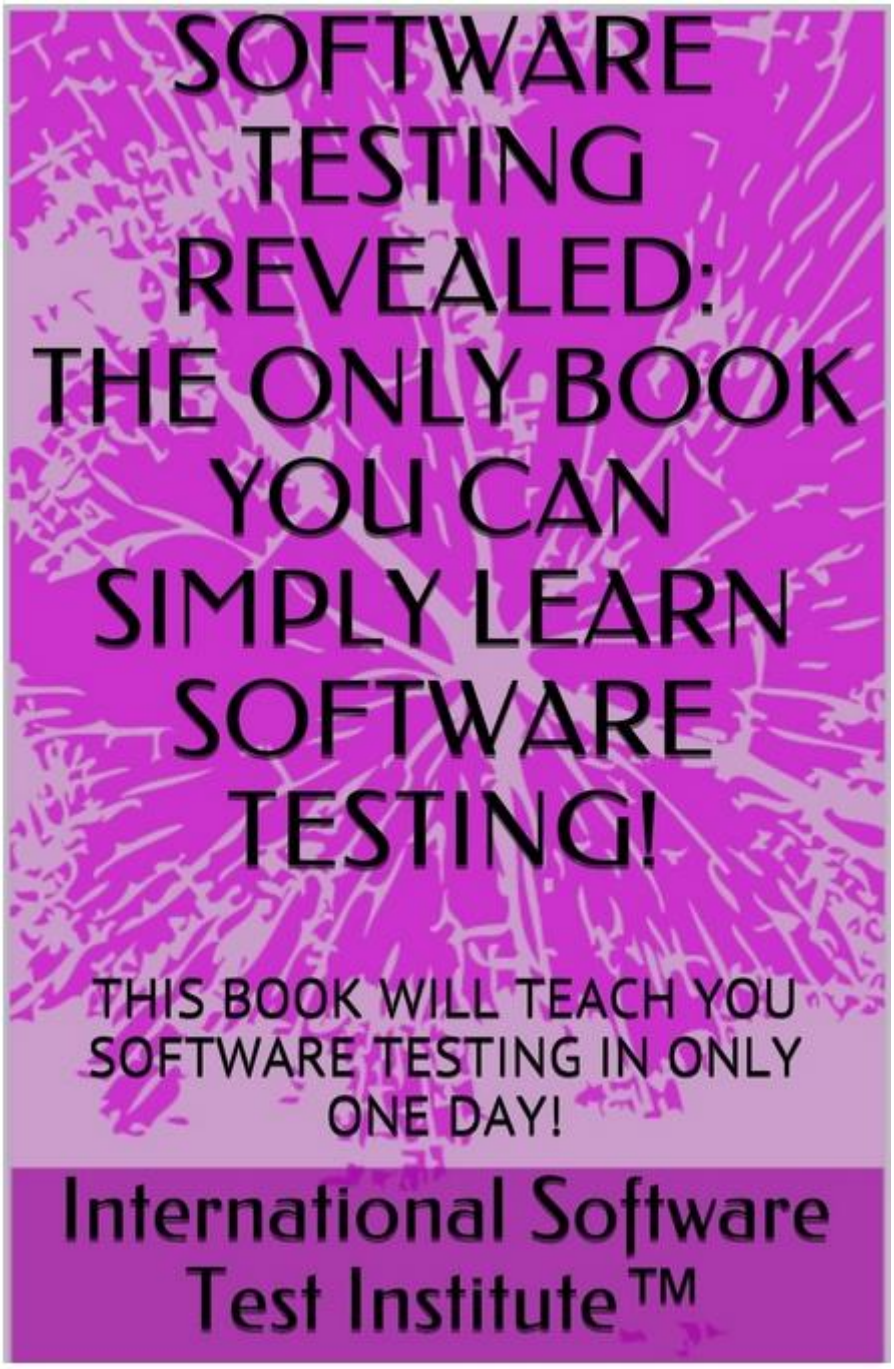

SOFTWARE TESTING REVEALED: THE ONLY BOOK YOU CAN SIMPLY LEARN SOFTWARE TESTING!
THIS BOOK WILL TEACH YOU SOFTWARE TESTING IN ONLY ONE DAY!
International Software Test Institute™

Contents

ABOUT THE AUTHOR

International Software Test Institute™ is an independent Institute which helps Organizations and Professionals worldwide prove their competence and knowhow in Software Testing and get them certified with our Software Testing Certification Programs. Our Accredited Software Tester, Accredited Software Test Manager and Accredited Software Test Automator Certification Programs have proven their worldwide Acceptance and Reputation by being the choice of more than 297'000 Software Testing Practitioners in 143 Countries.

Software Testing is an open process which can be combined with other Software Engineering Processes and Frameworks, and yet before International Software Test Institute was established, there used to be no reasonable way for Software Testing and Software Quality Assurance Professionals to obtain Software Testing Certifications and to prove their competence in Software Testing domain. Software Testing Professionals had to pay expensive fees for the money-driven Software Testing Certification Programs of other Certification Entities.

International Software Test Institute aims to remove the barriers set in front of the Software Testing Professionals in developed and emerging markets by saving them from paying unreasonable fees for Software Testing Classroom Trainings and Software Testing Certification Examinations before they certify their knowhow in Software Testing and Software Quality Assurance.

International Software Test Institute provides three major Online Software Testing Certification Programs which are designed by our consortium of renowned Software Testing Experts and Coaches participated from all major Industries. These certification programs are:

- Accredited Software Tester Certification Program (ASTC™)
- Accredited Software Test Manager Certification Program (ASTMC™)
- Accredited Software Test Automator Certification Program (ASTAC™)

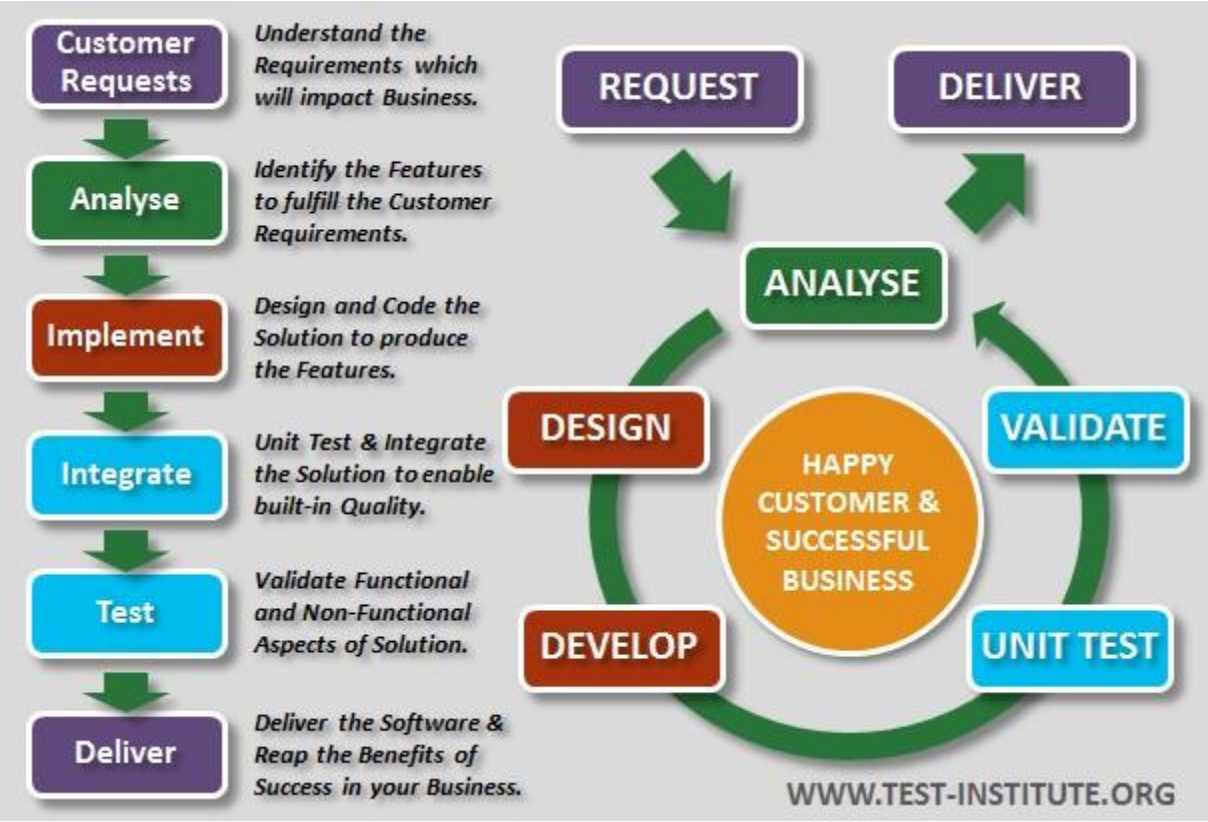
Our one-of-a-kind industry leading registration, examination and certification process is very simple, quick and completely online. You can find all details under the following link:
http://www.test-institute.org/Accredited_Software_Test_Manager_Certification_ASTMC_Program.php

Afterwards, please feel free to do your registration from the following link:
http://www.test-institute.org/Register_Software_Testing_Certification_Program.php

All the Best and Happy Reading!

1. Introduction To Software Testing

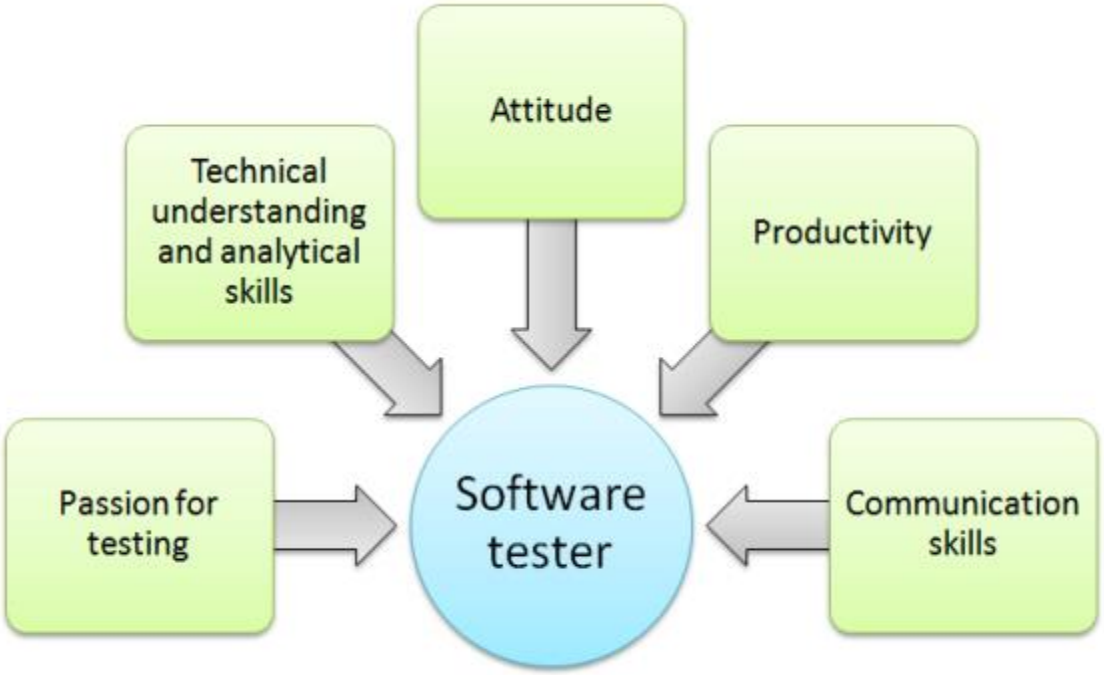
Software testing is nothing but an art of investigating software to ensure that its quality under test is in line with the requirement of the client. Software testing is carried out in a systematic manner with the intent of finding defects in a system. It is required for evaluating the system. As the technology is advancing we see that everything is getting digitized. You can access your bank online, you can shop from the comfort of your home, and the options are endless. Have you ever wondered what would happen if these systems turn out to be defective? One small defect can cause a lot of financial loss. It is for this reason that software testing is now emerging as a very powerful field in IT.



Software Testing Methodology in Software Engineering

Although like other products software never suffers from any kind of wear or tear or corrosion but yes, design errors can definitely make your life difficult if they go undetected. Regular testing ensures that the software is developed as per the requirement of the client. However, if the software is shipped with bugs embedded in it, you never know when they can create a problem and then it will be very difficult to rectify defect because scanning hundreds and thousands of lines of code and fixing a bug is not an easy task. You never know that while fixing one bug you may introduce another bug unknowingly in the system.

Software testing is now a very significant and integral part of software development. Ideally, it is best to introduce software testing in every phase of software development life cycle. Actually a majority of software development time is now spent on testing.



Introduction To Software Testing

So, to summarize we can say that:

1. Software testing is required to check the reliability of the software
2. Software testing ensures that the system is free from any bug that can cause any kind of failure
3. Software testing ensures that the product is in line with the requirement of the client
4. It is required to make sure that the final product is user friendly
5. At the end software is developed by a team of human developers all having different viewpoints and approach. Even the smartest person has the tendency to make an error. It is not possible to create software with zero defects without incorporating software testing in the development cycle.
6. No matter how well the software design looks on paper, once the development starts and you start testing the product you will definitely find lots of defects in the design.

You cannot achieve software quality without software testing. Even if testers are not involved in actual coding they should work closely with developers to improve the quality of the code. For best results it is important that software testing and coding should go hand in hand.

Software Testing Overview

Defects arise in software due to many reasons. As a matter of fact it is said that everysoftware application has some defects embedded in it but not every defect is a threat to the system. There is a lot that can be accomplished with the help of software testing. Testing helps in evaluating the quality of software.

There are many reasons why software testing has gained so much of importance in the field of information technology. Firstly, testing helps in reducing the overall cost of the software development project. If testing is ignored in the initial development stages to save a small amount of money then it may turn out to be a very expensive matter later because as you move on with development process it becomes more and more difficult to trace back defects and rectifying one defect somewhere can introduce another defect in some other module.

The requirement is finalized after several discussions with the client. Testing ensures that the software behaves and looks exactly like what is mentioned in the requirements specification document, so that when software is delivered to the client there are no arguments about the variation from the original requirements. Software testing helps in strengthening the market reputation of a company. Well tested software is of good quality and good quality means better feedback and reviews.

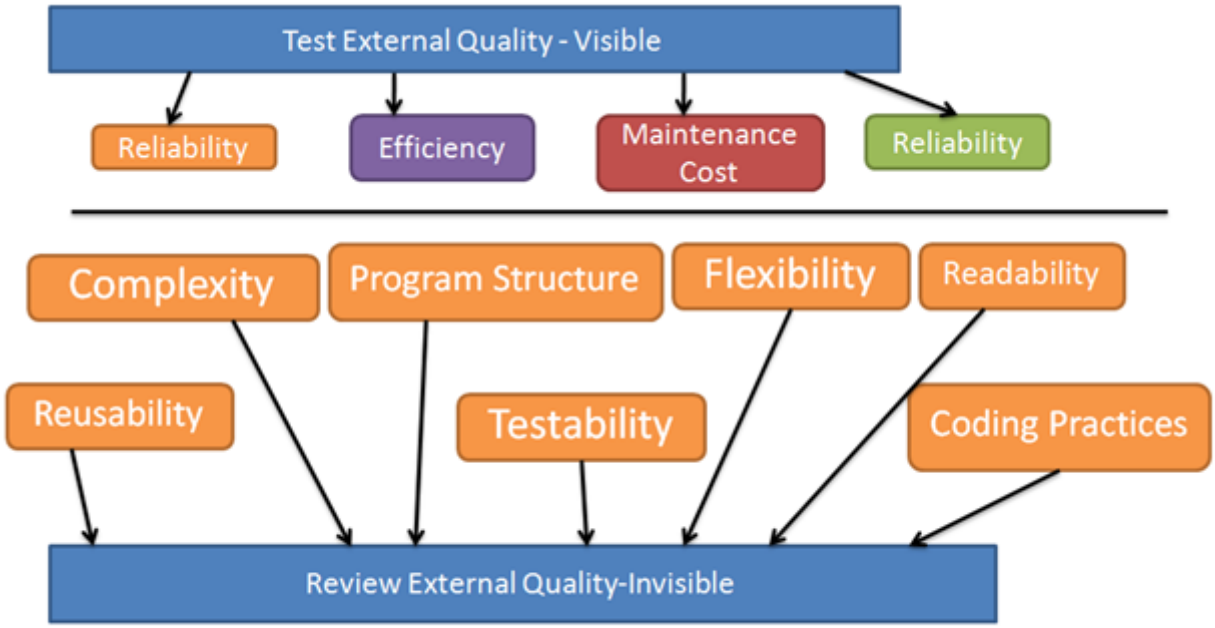
In order to achieve best results it is important to organize all your testing efforts and this is what this Software Testing Training provided by International Software Test Institute is all about. Software testing cannot be fruitful without proper planning. To live up to the expectations of the client it is important to plan every step carefully. A lot of things need to be considered in order plan your testing efforts. Software testing should be planned keeping budget, schedule and performance in mind in order to achieve best results.

All testing activities require planning. It is important to outline a test plan that will give in details about how each activity will be carried out. Test plan is also required to ensure that all aspects of the software are covered thoroughly and there is no repetition of testing process so that time and effort is not wasted.

The latest trend now is to involve the testing team in specification writing process. It is important that the testing team understands the requirements of the client clearly as the entire development is based on the requirement defined by the client. Anything that is not in line with the requirement is a defect. So, the testing team should have a clear idea about what the final outcome of running software should be like. As a matter of fact it is important to start writing test cases in parallel to specification writing. This will help the testers analyze whether all the requirements are testable or not. When you write test cases in parallel to specification writing process you will think critically about the specifications and you will know if there is an issue with the requirement or if there is something that cannot be developed.

2. What is Software Quality Assurance?

When we talk about software quality, we are actually talking about the evaluation of the software based on certain attributes. A software quality is defined based on the study of external and internal features of the software. The external quality is defined based on how software performs in real time scenario in operational mode and how useful it is for its users. The internal quality on the other hand focuses on the intrinsic aspects that are dependent on the quality of the code written. The user focuses more on how the software works at the external level, but the quality at external level can be maintained only if the coder has written a meaningful good quality code.



Software Quality

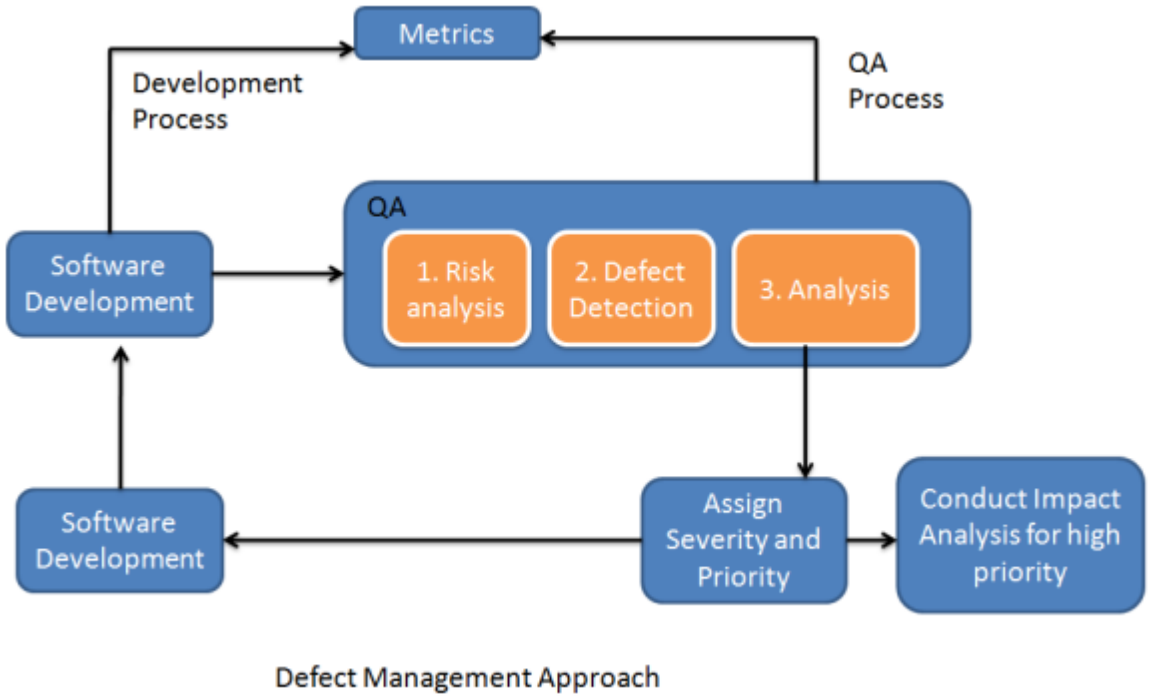
What Is Software Quality Assurance?

Presently there are two important approaches that are used to determine the quality of the software:

1. Defect Management Approach
2. Quality Attributes approach

As mentioned before anything that is not in line with the requirement of the client can be considered as a defect. Many times the development team fails to fully understand the requirement of the client which eventually leads to design error. Besides that, the error can be

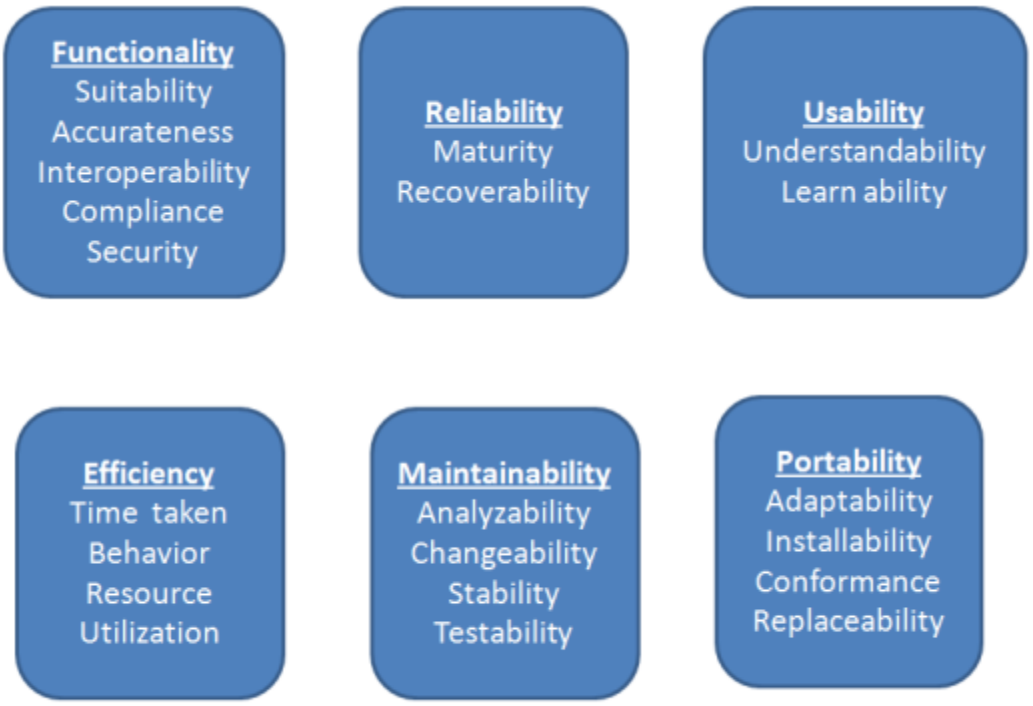
caused due to poor functional logic, wrong coding or improper data handling. In order to keep a track of defect a defect management approach can be applied. In defect management, categories of defects are defined based on severity. The number of defects is counted and actions are taken as per the severity defined. Control charts can be created to measure the development process capability.



Defect Management Approach

Defect Management Approach

Quality Attribute Approach on the other hand focuses on six quality characteristics that are listed below:



Quality Attribute Approach

Quality Attributes Approach

1. Functionality: refers to complete set of important functions that are provided by the software

- Suitability: whether the functions of the software are appropriate
- Accurateness: are the functions implemented correctly?
- Interoperability: how does the software interact with other components of the system?
- Compliance: is the software in compliance with the necessary laws and guidelines?
- Security: Is the software able to handle data related transaction securely?

2. Reliability: this refers to the capability of software to perform under certain conditions for a defined duration. This also defines the ability of the system to withstand component failure.

- Maturity: Frequency of failure of software
- Recoverability: this gives an idea of a system's ability to get back into full operation after failure.

3. Usability: refers to the ease of use of a function.

- Understandability: how easily the functions can be understood
- Learn ability: How much effort the users of different level need to put in to understand the functions.

4. Efficiency: generally depends on good architecture and coding practices followed while developing software.

5. Maintainability: also known as supportability. It is greatly dependant on code readability and complexity and refers to the ability to identify and fix a fault in a software:

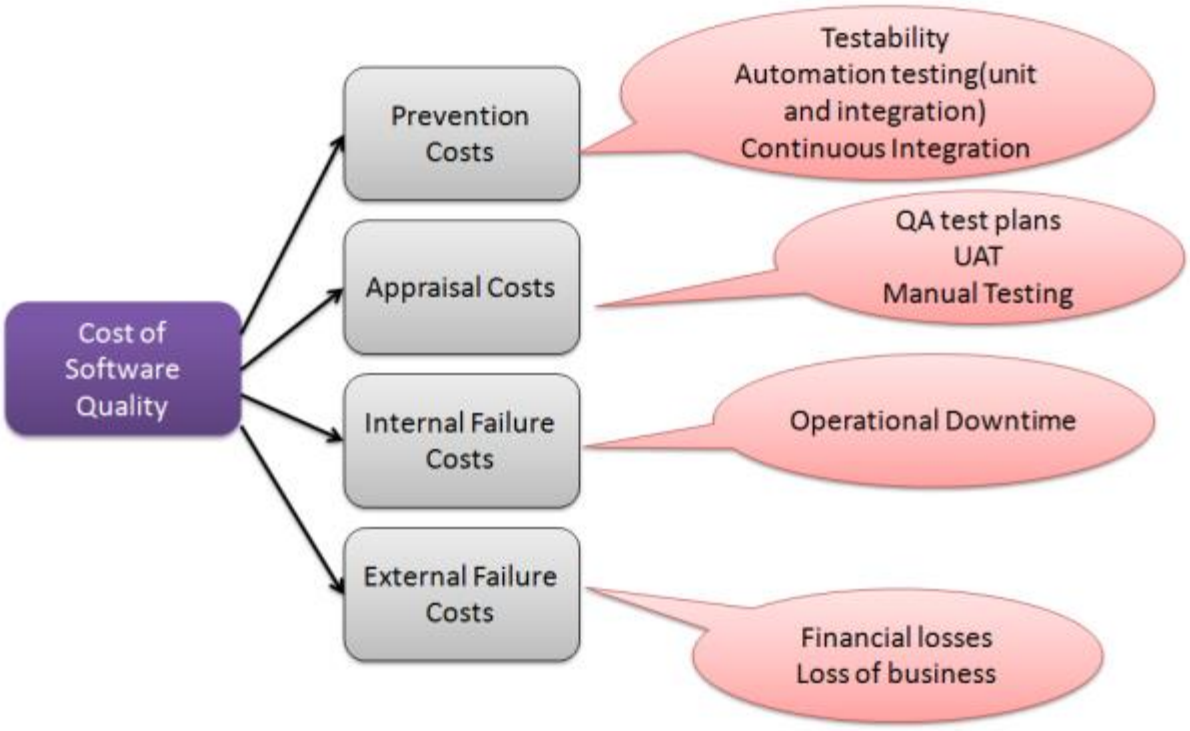
- Analyzability: identification of the main cause of failure.
- Changeability: defines the effort that goes in modification of code to remove a fault.
- Stability: how stable a system is in its performance when there are changes made to it
- Testability: how much effort goes in testing the system.

6. Portability: Ability of the system to adopt to changes in its environment

- Adaptability: how easily a system adapts to the changes made in specifications
- Installability: how easily a system can be installed.
- Conformance: this is same as compliance in functionality.
- Replaceability: how easy it is to replace a component of the system in a given environment.

Cost of Software Quality

Cost of quality is important because when you decide to conduct software testing for your product you are actually going to invest your time, money and effort in getting quality checks done. By conducting an analysis of cost of software quality you would know what the return on that investment (ROI) is.



Cost of Software Quality

Cost of quality is calculated by analyzing the conformance costs and non conformance costs. A conformance cost is related to:

1. Prevention costs: amount spent on ensuring that all quality assurance practices are followed correctly. This includes tasks like training the team, code reviews and any other QA related activity etc.
2. Appraisal costs: this is the amount of money spent on planning all the test activities and then carrying them out such as developing test cases and then executing them.

The non conformance cost on the other hand is the expense that arises due to:

1. Internal failures: it is the expense that arises when test cases are executed for the first time at internal level and some of them fail. The expenses arise when the programmer has to rectify all the defects uncovered from his piece of code at the time of unit or component testing.
2. External failures: it is the expense that occurs when the defect is found by the customer instead of the tester. These expenses are much more than what arise at internal level, especially if the customer gets unsatisfied or escalates the software failure.

Cost of Software Failure

We know that a software failure is caused when:

1. It displays lack of ability to keep up: this generally happens when the software starts aging. As it grows old the size increases because the easiest way of adding a feature is by adding new code without touching any part of code written earlier. Over a period of time it becomes bulky and it becomes difficult to identify the sections of code that need to be changed.
2. Performance drop is observed: Every application generally slows down with age and tends to occupy more and more computer memory therefore it is better to switch to other software.
3. It doesn't seem to be reliable: It is a known fact that every time when changes are made to the code of the software to fix an error, more defects are introduced in the system. Surprisingly, this is one of the major reasons for increased failure rates and in order to save situation it is always better to ditch the project or give up bug fixing.

Software Testing VS Quality Assurance

In IT industry it is often observed that people generally don't differentiate between the software quality assurance and software testing. Testers are often looked upon as Software Quality Assurance professionals because the objectives of software testing as well as quality assurance are the same .i.e. to ensure that the software is of top quality.
As the name suggests quality assurance processes are carried out to assure the quality of the product is in line with the requirement of the client. The quality assurance professionals work on development and implementation of all the necessary processes to ensure that all the necessary procedures of software development lifecycle are followed correctly. Quality assurance is a proactive activity that is focused on:

1. Defect Prevention
2. Processes
3. Continuous improvement of this processes

Software testing on the other hand is carried to identify or uncover defect and errors in the software. It involves actual rigorous testing of the software to see if there are any defects or variations from the client's requirement that needs to be fixed. Software testing is a part of quality control process and it focuses only on product oriented activities. Software testing is carried out during the testing phase and only defects are identified and not corrected in this process. Fixing defects is not a part of software testing.

Quality Assurance VS Quality Control

Another subject that is closely related to quality assurance is quality control. People often get confused between the two but there is a huge difference. While quality assurance is all about preventive activities, quality control focuses on corrective processes.
Here is what you need to understand: software testing is a subset of quality control and quality control is a subset of quality assurance. Theentire focus of Quality assurance is on implementation of processes and procedures that are required for the verification of the software under development and the requirements of the client.

Quality Assurance VS Quality Control

Quality control on the other hand deals with actual activities that ensure that the product is being developed as per the defined requirements. It deals with all the actions that are important to control and verify certain characteristics of the product including testing. Examination and testing of the products is the most important aspect of quality control.

Companies employ quality control team to identify if there is any product or service that does not meet the company's standard of quality. If there is an issue the quality control team has the authority to stop the production of that product till the issue is resolved.

Importance of Audit and Inspection

Audit comprises of some very systematic processes that define how the software testing is taking place in the organization. The audit team examines all the processes that are conducted at the time of testing. IEEE defines audit as a review of documented processes to ensure that the organization or a team is following all the processes as per the defined standards.

Inspection can be a formal or an informal review of software requirement, designer or code. It is conducted by a team or an individual person other than the author to check if there are any violations or deviations from the defined development standards. The following processes are considered as part of Inspection:

1. Planning
2. Overview Preparation
3. Inspection Meeting
4. Rework
5. Follow up

3. What Is Software Testing?

So, finally we come to the main topic that is software testing itself. You have already understood the meaning of software testing and why it is important while going through the previous sections. Here, from this section onwards we will take an in-depth look at the subject, but before we move on let's just revise the definition of software testing.

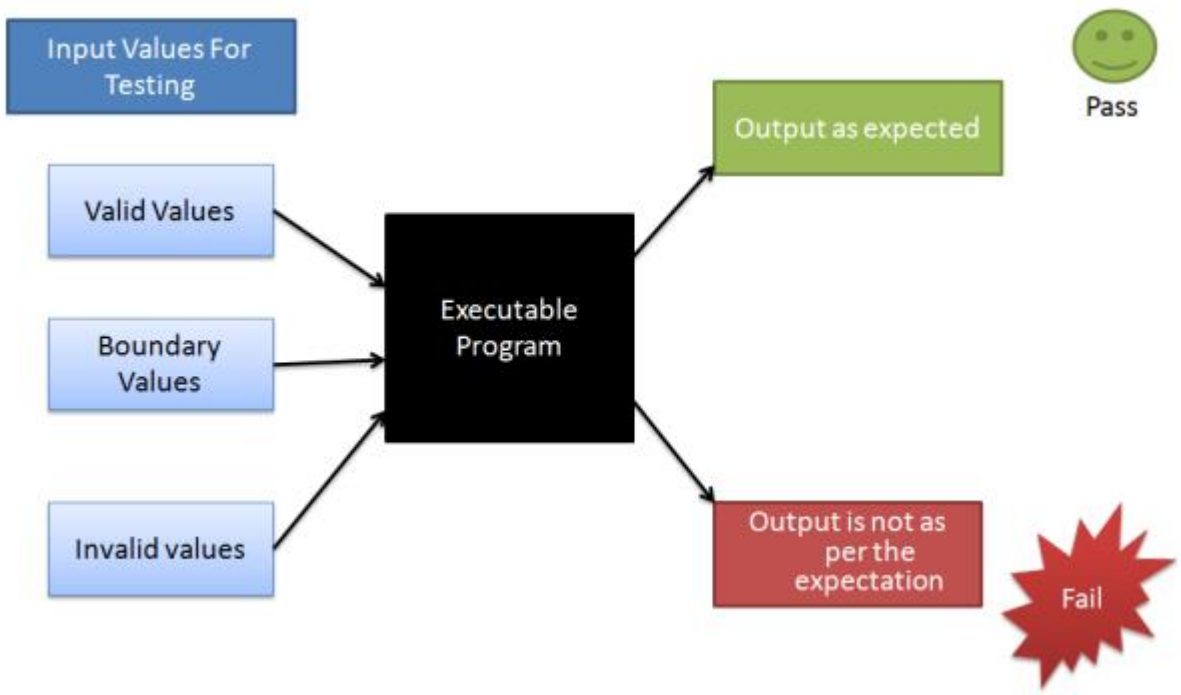
Software testing is nothing but the process of assessing the functionality of software to ensure that it is in line with the requirements of the customer. Testing is broadly classified into:

1. Dynamic Testing: carried out by executing the program
2. Static Testing: involves examination of code and related documents.

Dynamic and static testing is often used together.

Black Box Testing

Black box testing focuses only on the functionality of the software. The tester does not look into the internal details of the software. Black box testing is carried out at all levels of software testing – unit, integration, system and acceptance.

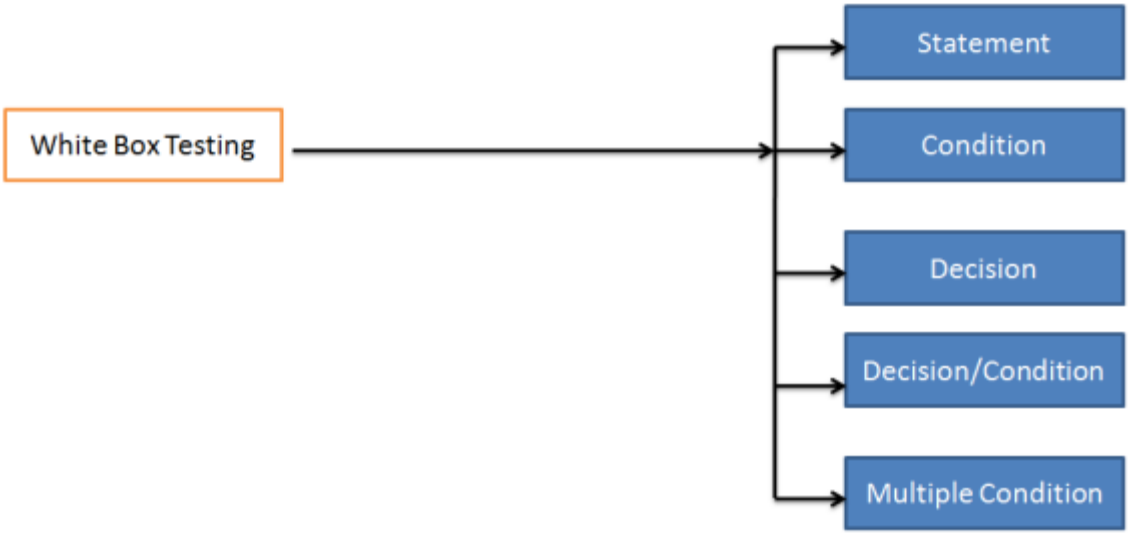


Black Box Testing

The testing procedures for black box testing are very simple. The tester only focuses on what the software is supposed to do. The tester is not supposed to focus on how the software is managing the function internally. The test cases for black box testing are created keeping only the specifications and requirements in mind. No test case is created to check the internal logic of the software. The tester just feeds in valid and invalid inputs and checks the output for these values.

White Box Testing

Unlike black box testing, white box testing is carried out in depth to the level of the source code. In this form of testing the internal logic, its implementation and working is examined and the test cases are written to check the how the software is working at the internal level. White box testing can be carried out at the level of unit, integration and system level. White box testing is often used to detect internal design errors which are otherwise very difficult to uncover however, this form of testing does not check for missing requirements or specifications.



White Box Testing

Statement Coverage

Statement coverage is a form of testing in which code is tested in such a way that each statement of the code is executed at least once. The idea behind this form of testing is to ensure that every statement in every block of code is executed at least once and the results are observed. This form of testing is also referred to as line coverage or segment coverage form of testing.

The point to be noted here is that every statement is executed once which means that there may be some conditions in some blocks that may not get tested in this manner. Therefore there is a possibility that some errors may go undetected in statement coverage process.

Decision Coverage

Decision coverage or branch coverage deals with testing of all true and false conditions of the code. The reason why it is also called branch coverage is because a branch is an outcome of decision. It is considered to be a more effective form of testing than simple statement coverage. A decision statement can be:

1. An IF statement
2. A loop control statement such as do-while
3. A statement that can have two or more outcomes also known as CASE statement.

The good thing about decision coverage is that you are able to validate all branches in the code and it is able to check the efficiency of the code in a better manner than statement coverage approach.

Condition Coverage

Condition coverage testing is carried out to check conditions which are generally Boolean expressions and provide result in TRUE or FALSE. Condition coverage may or may not cover the entire decision coverage. In this process only those conditions that return true or false are tested. The expressions that returns a Boolean condition generally plays a very important role in the final decision. This is the reason why condition coverage testing is carried out.

Decision / Condition Coverage

As the name suggests decision/condition coverage methodology includes testing all decisions and all the logical conditions with all possible scenarios that can generate a true or false outcome. It is considered to be a very strong way of testing software.

Multiple Condition Coverage

Multiple condition coverage or condition combination coverage is carried out to check output for multiple combinations of conditions. For example:

```
If (x=true OR y=true)
  Then
Print (“Hello”)
  Else
Print (“Bye”)
```

Here, in this case there are four possible condition combinations:
Testcase1: x=true; y=true
Testcase2: x=true; y=false
Testcase3: x=false; y=true
Testcase4: x=false; y=false

Similarly, for 3 expressions there will be 8 combinations of conditions.

4. Fundamentals of Software Testing

Software testing is a vast subject. There are software applications and system engineered for numerous domains and industries, and for a tester, every testing project is a new challenge because he has to understand the client's point of view and the domain before moving on with testing activities. From project to project, a tester may have to change the testing methodologies as well. It is therefore very important to keep the fundamentals right. Getting the fundamentals right in the first place is biggest prerequisite to become successful in software testing.

Why Software Testing Is Necessary?

An error, defect or a bug can be caused by developers. It is not intentional but considering the complexity with which various software are being developed these days, it is quite possible for a developer to misunderstand and implement wrong logic and produce wrong code.

Testing is necessary because it helps us in identifying the faults in software. Once these defects have been detected they can be easily rectified and quality of the software can be improved. So, software testing is necessary so that bug free applications can be developed and delivered. When a company decides to develop software for a client there are certain legal, contractual and industry-specific requirements based on the deal is made. A quality conscious company will definitely include software testing in its best practices.

It is difficult to say how much testing is enough but the fact is that if testing is planned carefully and good test cases are made then it is very much possible to deliver high quality software.

Who Does The Software Testing?

There is often a debate on who should actually test the software. People often question that why developers are not allowed to test. Well, a developer generally checks his code several times before he submits it for testing and still in most cases it is never error free because a developer is generally blind to his own mistakes.

A tester on the other hand looks at software from the point of view of the client. He is unbiased and his focus is only on the specifications and the requirements. So, a tester is able to look into areas that a developer may have ignored. So, the testing should always be carried out by independent testers.

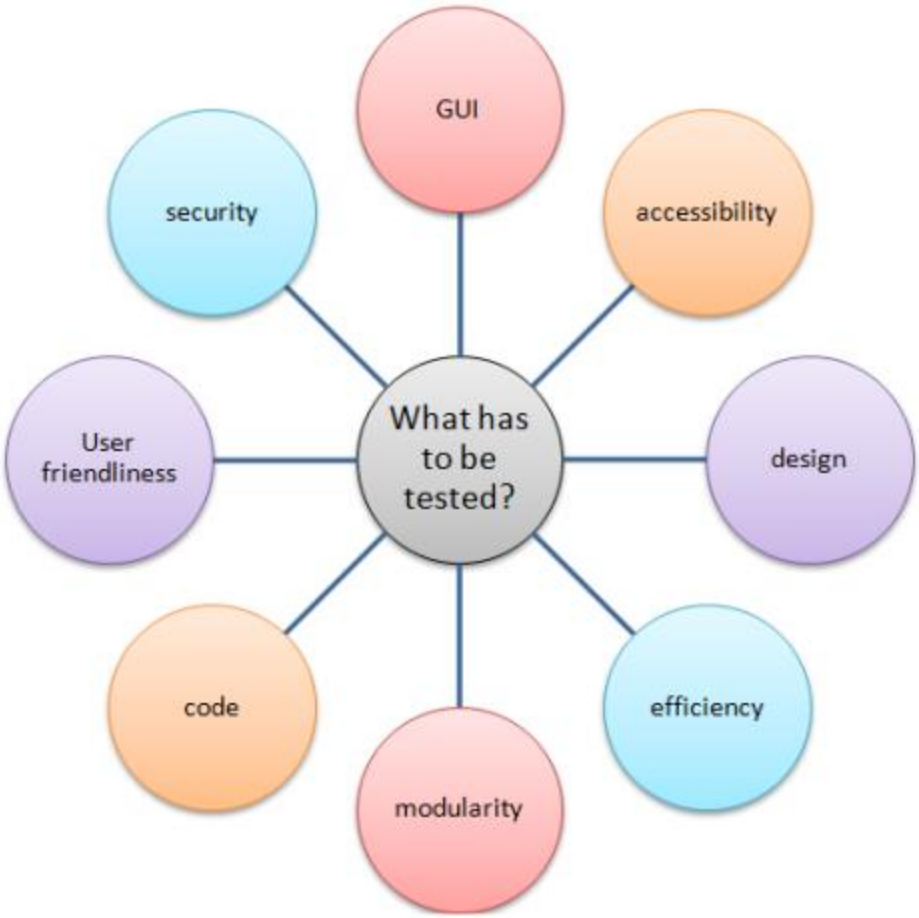
This approach does have some disadvantages. When the development and testing teams are different there is often a communication gap and sometimes, developers become careless towards coding and do not revise their code because they think that it is all a tester's job thereby increasing the burden on the tester.

Many times developers share their work amongst each other and test each other's work. This is known as buddy testing. Every development team should have dedicated testers and every project generally has at least one dedicated testing team. Some companies believe in having separate

teams for different types of testing, this means different teams for usability, performance, security and other forms of testing. Some companies believe in outsourcing software testing work which means they hire a firm or independent testers or consultants to have a look at and test their project.

What Has To Be Really Tested?

The tester should have a good understanding about the project requirements. A fair idea about the real time scenario where the software will be implemented can help the tester understand how to carry out testing for the project. It is very important to know what has to be really tested in order to devise a testing strategy.



What Has To Be Really Tested?

When Is The Software Testing Done?

The earlier the testing team starts testing the software the easier it would be for the developers to complete the project on time and this would also save a lot of time, money and effort. Starting

testing in the later stages of development can turn out to be an expensive matter as it is very difficult to rectify defects once the software has reached the final stages of development. Dividing software development into stages and then testing work done in every stage before moving on to the next stage helps in finishing the software development in time with good results. This also helps in better integration of different modules because you already know that every module has been tested independently and is working as per the given specifications.

How Often Do We Need To Test?

How often you need to test depends on how important the quality is for you. Ideally, testing should go hand in hand with development and a tester should focus on discovering maximum number of defects during the initial phases of software development so that if the design of the software requires any changes then it can be done early as it will be very difficult and expensive to make major changes in the project during the later stages of development.

How Often Do We Need To Test?

What Are Software Testing Standards?

Software testing standards are of great importance from the consumer's as well as producer's point of view. A consumer invests in the software and if the software is of good quality then at the end of the day he is satisfied that he has purchased the right thing for himself.

All reputed companies ensure that the software quality of product is governed by some sets of standards that have been approved by the public. By abiding by these standards a company gives assurance about its products and it is able to give guarantee to its customers only when it has followed some standards and knows that the software will behave in a certain manner. So, consumer knows that he is buying the right thing if it is of right standard.

From a producer's point of view standards help in improving the quality of the final product. Once a company has finalized the standard that it has to follow then it becomes easier for them to work on other software projects and whenever they start a new project they do not need start everything from scratch.There are many types of software testing standards defined for evaluating quality of software which can greatly improve the effectiveness of software testing however it is believed that till now no such standards have been made that can cover all aspects of software testing.

Standards that emphasize on having testing as part of larger requirement or standards supporting software testing are the ones that can be of use for software testing.

What Is Software Testability?

Software testability is used to measure how easily a software system can be tested. Testability is calculated in the early phases of software development in order to find out how many resources will be needed to finish of the testing process. While testing helps in uncovering defects, testability plays a significant role in identifying the key areas where bugs remain hidden from a tester's view. When testability is high it means that testing is easier and lower testability means that the testing effort should be increased. Testability can be determined by:

1. Controllability: Testing process can be optimized only if we can control it.
2. Observability: What you see is what can be tested. Factors affecting the final outcome are visible.
3. Availability: In order to test a system we have to get at it.
4. Simplicity: When the design is self-consistent, features are not very complex and coding practices are simple then there is less to test. When the software is not simple anymore it becomes difficult to test.
5. Stability: If too many changes are made to the design now and then there will be lot of disruptions in software testing.
6. Information: Efficiency of testing greatly depends on how much information is available for the software.

Here is what you need to know about testability:

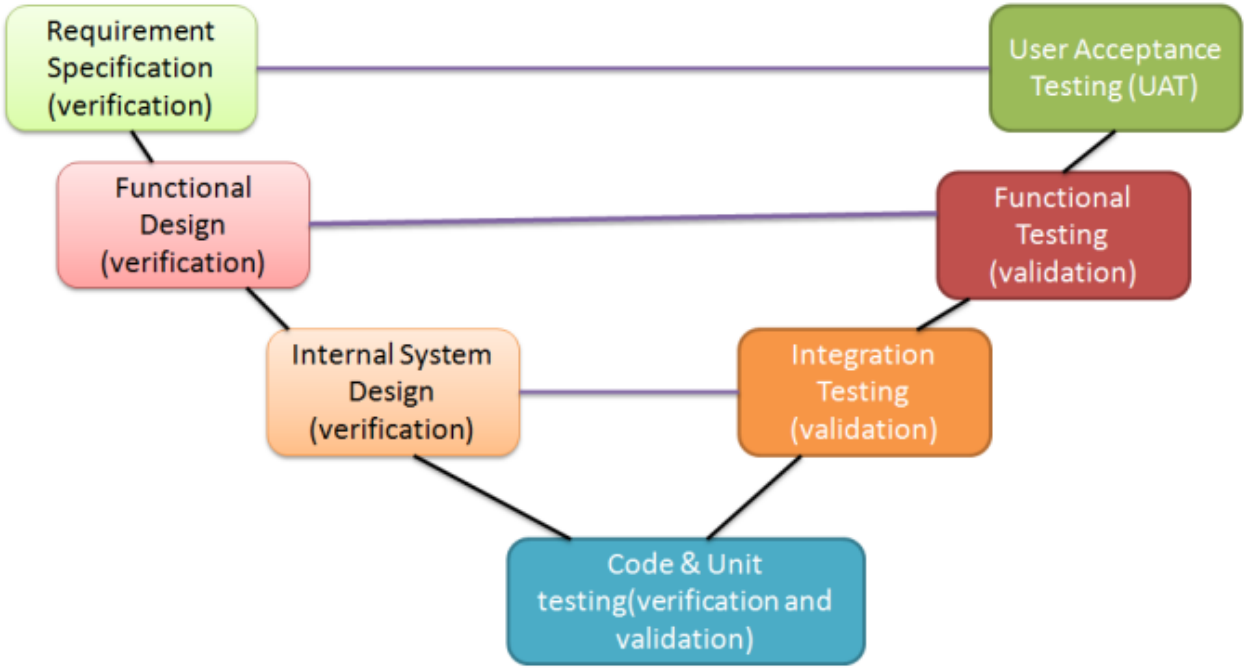
1. Higher testability means better tests and less amount of bugs
2. Lower testability means the tests are not of great quality and there is a possibility of more bugs in the system

Software Verification VS Software Validation

Verification and validation are two very important terms in software testing. People often get confused between the two however these two terms are related to two different types of analysis. Validation is helps in building the right system. While carrying out validation we look at whether the system is in line with customer's requirement or not.

Verification on the other hand helps in ensuring if the system is being developed in the right way. The focus of verification is on the quality of software that is being developed, whether it follows all the standards or not, is it well engineered?

So while validation checks if the specifications have been designed correctly to meet the customer's requirement, the verification checks if the software has been developed as per the software quality standards and norms of the software engineering organization.



Software Verification VS Software Validation

Some theories suggest that verification is carried out in every phase of software development lifecycle but the same is not the case with validation. Validations are crucial in the beginning and towards the end of the project, i.e. during the requirement analysis and acceptance testing. This tactic is not fully correct and almost impossible to follow. The actual fact is that until today it has been observed that it is very difficult to capture the entire set of client requirements during the beginning of the project. Software requirements often undergo several changes even after the development has started. Many times the changes are requested by the development team itself. It is therefore important to carry out validation and verification processes in every phase of software development.

Today, testers usually consider verification and validation also known as V&V as a powerful way of looking at various aspects of the software.

Software Testing VS Software Debugging

This is another topic where people generally get confused. Software testing and debugging may sound like one and the same thing but that is not actually the case. To start with, the process of debugging starts when software testing gets over. While software testing uncovers defects, debugging removes defects from the system.

Once testing is completed the testers submits the reports and the development team starts looking for the root cause of the defects. In this process the developer scans all the related modules and tries to find out the problem in the code; once that is done it is time to rectify the defect. So, debugging is the process in which the cause of reported defects is found out and then defects are rectified step by step.

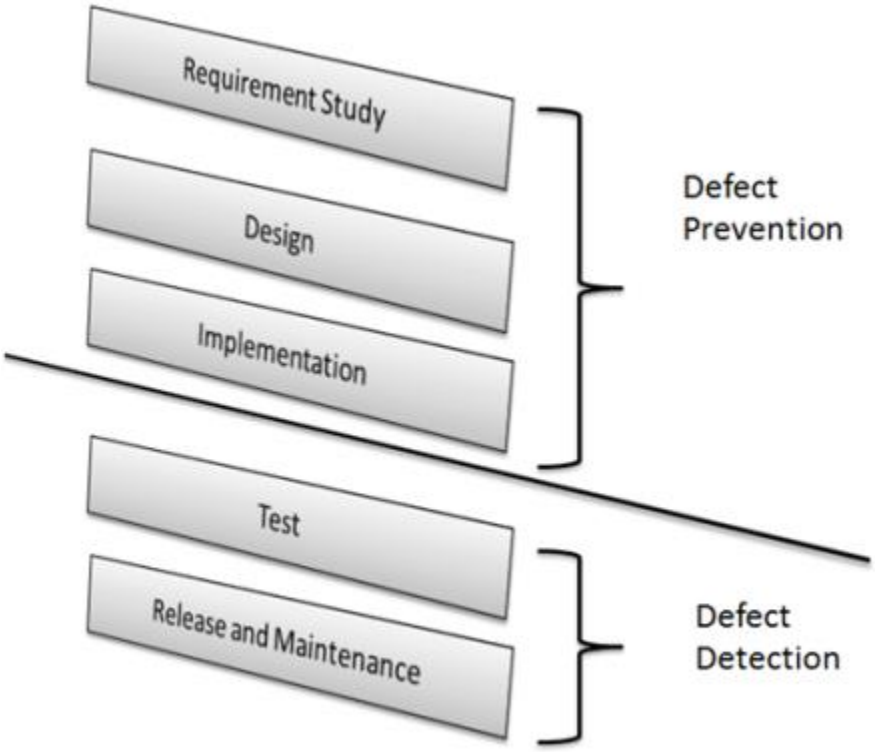
Debugging is the process of resolving existing issues. It not wise to rectify a defect till all possible reasons behind it is fully known. If you start debugging without complete knowledge about the defect then there are chances that in the process of rectifying one defect you may introduce some other in any of the interrelated modules. Developers should avoid experimentation at the time of debugging as this can cause a lot of problems. Once a defect is fixed the developer submits the work to the tester, who will thoroughly test the entire module.

1. Software testing uncovers defects and debugging locates and corrects it.
2. Software testing is a very important aspect of software development cycle whereas debugging is a result of testing activities.
3. Testing begins soon after development starts. Debugging starts when testers start reporting defects.
4. Software testing involves verification and validation (V&V) of the software whereas debugging looks in to actual cause behind the defect and corrects it.

What Is A Defect?

Defect is anything that is not in line with the software requirement specifications. Defect generally exists either in the code of the program or in its design. This results in incorrect output at the time of execution of the program.

1. A piece of code is called buggy when it has too many defects.
2. Bug reports the details about the nature of the bug.
3. Bug tracking tools are employed to track bugs in a system.
4. There is a very interesting testing technique in which defects are purposely injected into the code and then the outcome is monitored to check if the system works as expected in case of certain issues.



What Is A Defect?

What Are Severity And Priority?

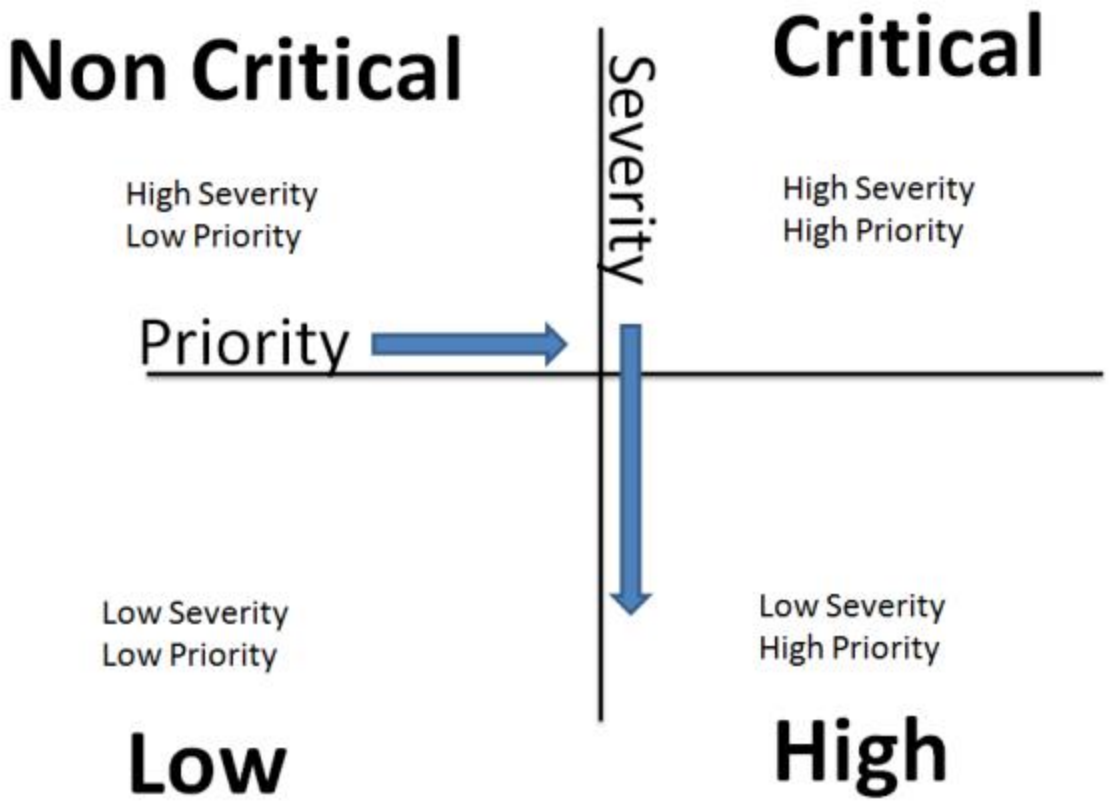
Severity defines how severe will be the impact of a defect on the performance of the system. The severity can be of one of the following types:

1. Critical: Such adefect does not allow the application to work properly due to system failure or corruption of data. Critical defects do not allow the user to move any further and puts them in a miserable position.
2. Major: The major defects are little less severe than critical defects. They can cause system to fail, however in case of major defect there is another possible way of achieving the desired result and the user need not get trained for this.
3. Moderate: These defects do not cause the system to fail but produce wrong or contradictory output.
4. Minor: Defects that do not cause system failure or affect the usability of the system and can be easily rectified are known as Minor defects.
5. Cosmetic: Defects related to the outlook or appearance of the system are called cosmetic defect.

When a defect is reported, the test report mentions priority along with the severity of the defect. Priority actually tells the developer the order in which defects should be resolved. It can be of the following types:

1. Low: the defect does not require immediate attention and should be rectified after the defects with higher priority have been resolved.
2. Medium: The defect should be resolved soon after the defects with higher priority have been resolved.
3. High: The defect with high priority means that it requires immediate attention and should be resolved as soon as possible.

So, once you are provided with the report of severity and priority of defects here is what you need to know:



What Are Severity And Priority?

1. If a defect has high priority and high severity, then it means that there is a problem in the basic functionality of the system and the user is not in a position to use the system. Such defects should be rectified immediately.
2. Defects having high priority and low severity can be something like spelling mistake in the company's name or issues with logo. Such defects are of low severity but must be rectified immediately and should be considered as high priority defect.

3. High Severity and low priority defect means that there is a major defect in some module but the user would not be using it immediately so the defect can be rectified a little later.
4. Low priority and low severity defects are generally cosmetic in nature and do not affect the functionality of the system hence such defects are rectified in the end.

5. Software Testing Roles and Responsibilities

In case of software testing every company defines its own level of hierarchy, roles and responsibilities but on a broader level, if you take a look you will always find the following two levels in a software testing team:

Test lead/manager: A test lead is responsible for:

- Defining the testing activities for subordinates – testers or test engineers.
- All responsibilities of test planning.
- To check if the team has all the necessary resources to execute the testing activities.
- To check if testing is going hand in hand with the software development in all phases.
- Prepare the status report of testing activities.
- Required Interactions with customers.
- Updating project manager regularly about the progress of testing activities.

Test engineers/QA testers/QC testers are responsible for:

- To read all the documents and understand what needs to be tested.
- Based on the information procured in the above step decide how it is to be tested.
- Inform the test lead about what all resources will be required for software testing.
- Develop test cases and prioritize testing activities.
- Execute all the test case and report defects, define severity and priority for each defect.
- Carry out regression testing every time when changes are made to the code to fix defects.

Overview Of Software Engineering Team

How a software application shapes up during the development process entirely depends on the how the software engineering team organizes work and implements various methodologies. For an applicationto develop properly, it is important that all processes incorporated during the software development are stable and sustainable. Many times developers come under pressure as the delivery date approaches closer this often affects the quality of the software. Rushing through the processes to finish the project on time will only produce a software application whichhas no or minimal use for the customers. Hence, work organization and planning is important and sticking to the plan is very important. The project manager should ensure that there are no obstacles in the development process and if at all there is an issue it must be resolved with immediate attention.

Overview Of Software Testing Team

How soon and how well you can achieve your testing goals depends solely on the capabilities of the testing team. Within the testing team itself it is important to have the correct blend of testers who can efficiently work together to achieve the common testing goals. While forming a team for testing, it is important to ensure that the members of the team jointlyhave a combination of all the relevant domain knowledge that is required to test the software under development.

It is very important to ensure that the software testing team has a proper structure. The hierarchy and roles should be clearly defined and responsibilities too should be well defined and properly distributed amongst the team members. When the team is well organized the work can be handled well. If every team member knows what duties he or she has to perform then they will be able to finish their duties as required well within the time limit. It is important to keep track of the testers' performance. It is very important to check what kind of defects the tester is able to uncover and what kind of detects he tends to miss. This will give you a fair idea about how serious your team is about the work.

All the team members should work together to prepare a document that clearly defines the roles and responsibilities of all the team members. Once the document is prepared the role of each member should be communicated clearly to everyone. Once the team members are clear about who is going to handle which area of the project, then in case of any issue it will be easy to determine who needs to be contacted.

Each member of the team should be provided with the necessary documents that provide information on how the task would be organized, what approach will be followed, how things are scheduled, how many hours have been allocated to each member and all details related to applicable standards and quality processes.

Software Tester Role

A Software tester (software test engineer) should be capable of designing test suites and should have the ability to understand usability issues. Such a tester is expected to have sound knowledge of software test design and test execution methodologies. It is very important for a software tester to have great communication skills so that he can interact with the development team efficiently. The roles and responsibilities for a usability software tester are as follows:

1. A Software Tester is responsible for designing testing scenarios for usability testing.
2. He is responsible for conducting the testing, thereafter analyze the results and then submit his observations to the development team.
3. He may have to interact with the clients to better understand the product requirements or in case the design requires any kind of modifications.
4. Software Testers are often responsible for creating test-product documentation and also has to participate in testing related walk through.

A software tester has different sets of roles and responsibilities. He should have in depth knowledge about software testing. He should have a good understanding about the system which means technical (GUI or non-GUI human interactions) as well as functional product aspects. In order to create test cases it is important that the software tester is aware of various testing techniques and which approach is best for a particular system. He should know what are various phases of software testing and how testing should be carried out in each phase. The responsibilities of the software tester include:

1. Creation of test designs, test processes, test cases and test data.

2. Carry out testing as per the defined procedures.
3. Participate in walkthroughs of testing procedures.
4. Prepare all reports related to software testing carried out.
5. Ensure that all tested related work is carried out as per the defined standards and procedures.

Software Test Manager Role

Managing or leading a test team is not an easy job. The company expects the test manager to know testing methodologies in detail. A test manager has to take very important decisions regarding the testing environment that is required, how information flow would be managed and how testing procedure would go hand in hand with development. He should have sound knowledge about both manual as well as automated testing so that he can decide how both the methodologies can be put together to test the software. A test manager should have sound knowledge about the business area and the client's requirement, based on that he should be able to design a test strategy, test goal and objectives. He should be good at project planning, task and people coordination, and he should be familiar with various types of testing tools. Many people get confused between the roles and responsibilities of a test manager and test lead.For a clarification, a test lead is supposed to have a rich technical experience which includes, programming, handling database technologies and various operating systems, whereas he may not be as strong as Software Test Manager regarding test project management and coordination. The responsibilities of the test manager are as follows:

1. Since the test manager represents the team he is responsible for all interdepartmental meetings.
2. Interaction with the customers whenever required.
3. A test manager is responsible for recruiting software testing staff. He has to supervise all testing activities carried out by the team and identify team members who require more training.
4. Schedule testing activities, create budget for testing and prepare test effort estimations.
5. Selection of right test tools after interacting with the vendors. Integration of testing and development activities.
6. Carry out continuous test process improvement with the help of metrics.
7. Check the quality of requirements, how well they are defined.
8. Trace test procedures with the help of test traceability matrix.

Software Test Automator Role

Software test automator or an automated test engineer should have very good understanding of what he needs to test- GUI designs, load or stress testing. He should be proficient in automation of software testing, and he should be able to design test suites accordingly. A software test automator should be comfortable using various kinds of automation tools and should be capable of upgrading their skills with changing trends. He should also have programming skills so that he is able to write test scripts without any issues. The responsibilities of a tester at this position are as follows:

1. He should be able to understand the requirement and design test procedures and test cases for automated software testing.
2. Design automated test scripts that are reusable.
3. Ensure that all automated testing related activities are carried out as per the standards defined by the company.

Interactions between Software Test Team And Business Teams

If at all a customer has any issues related to testing activities and operational matters of the project then it is the software testing manager who is responsible for communicating the details to the client regarding how things are being managed. The software testing manager not only answers the queries of the customers but also ensures that the project is completed on time as per the requirement of the customer.

Interactions between Software Test Team And Development Teams

In order to produce good software applications, it is important that software testing and software development teams work together with good understanding. For this it is important that the testers and developers are comfortable with each other's role and understand well that they have a common goal and it is wise to listen each other. A good communication skill is very important both for testers and developers.

Before getting started with testing work it is important to discuss the basic guidelines and expectations so that there is no confusion in later stages. Criticism should be taken in a positive sense. It is important to understand that developers and testers have a common goal of producing high quality software. A tester is not discovering bugs to show someone down, the idea is to learn from mistakes and avoid repeating them in future. A culture of constructive criticism can be of great help.

Interactions between Software Test Team And Release Management Teams

The release management teams are responsible for moving the software from development into production. This team is responsible for planning the releases for hardware, software and testing. It is also responsible for development of software development procedures and for coordinating interactions and training of releases. Software testing is considered to be a very important aspect of software engineering life cycle but it does not get over with development. Testing and verification is a very important part of release management exercise.

Interactions between Software Test Manager And Software Project Manager

The job of a software test manager is not an easy one. He has to recruit testing team and take responsibility for getting them trained. A software manager has to perform ongoing analysis of various testing processes and ensure that the testing team is carrying out all the processes correctly. This job is of great responsibility as the software testing manager is the one who

selects, introduces and implement various tools for testing. A software test manager is responsible for finalizing templates for testing documents, test reports and other procedures.

Since a software tester manager has to deal with all the details of various testing activities, it is very important for him to be in constant touch with the project manager and provide necessary support in project planning and scheduling so that the project can be successfully completed in time within the specified financial budget limits.

6. Software Testing Methods

We have discussed black box and white box testing techniques in section 3. Here we take a look at some more testing methodologies:

- Grey Box Testing
- Incremental Testing
- Thread Testing

Grey Box Testing

When black box testing (3.1) methodologies and white box testing methodologies(3.2) are used in a combination for software testing then it is called gray or grey box testing. This form of testing will look into logical as well as functional aspects of the software. In this form of testing the tester has little and not in-depth knowledge about what the code is supposed to do. Ideally, the tester should know about the internal data structures and algorithms.

Grey box testing is carried out when there is a need to test both sides of the software(functional and internal in one go). The tester should be able to write test cases that can test the software thoroughly with the help of data that can check all possible internal logic as well.

Grey box testing is a very intelligent form of testing where the tester is expected to know the functionality, architecture of the software, how data would flow and how exceptions would be handled. For big size projects it is always better to incorporate automation testing to check functionality and the user interface of the application as this would save a lot of time and the tester will not feel too stressed.

Incremental Testing

Incremental testing comes into picture once the tester has completed unit testing. This form of testing is used in integration of testing where there is need to check how various independent modules interact amongst themselves. In this form of testing the developers are asked to integrate the modules systematically using stubs or drivers. This form of integration testing methodology is referred to as incremental testing. Incremental testing can be classified into three types:

1. Top down integration: In this case the integration of the modules is done from top to bottom. All those modules that are unavailable are replaced by stubs.
2. Bottom up integration: In this case the integration of the modules is done from bottom to up. All the modules that are not available are replace by drivers.
3. Functional incremental: this form of testing takes place as per the functional specification documents. The integration of the modules and their testing takes place as per the defined functional specification.

The main advantage of incremental testing is that it can help in uncovering defects very early in time. The disadvantage is that development of stubs and drivers can take time.

Thread Testing

Thread testing methodology is used in testing applications based on client server architecture. A thread is the smallest unit of work that can be carried out by the system. During the initial stages of integration of a system it is very important to know if the system will be able to carry out the required functional tasks as per the requirement. It is very important to check if the software will be able to carry out all transactions as per the requirement. The ideal way of carrying out thread testing is to integrate threads incrementally first at subsystem level and then at the system level and then tested.

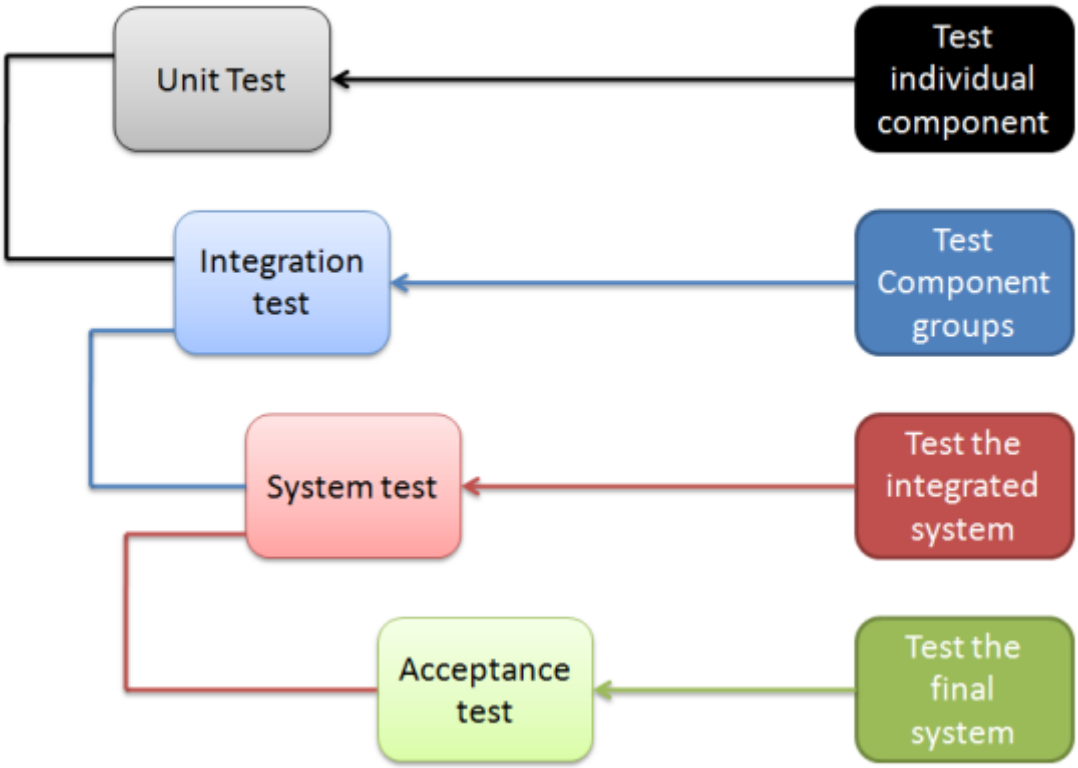
7. Software Testing Levels

Software testing has various levels. However, on broader scale software testing can be categorized into (1) Functionaltesting and (2) Non-functional testing. These topics will be discussed in detail.

Overview of Software Testing Levels

Here we will understand various levels of testing, namely:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing



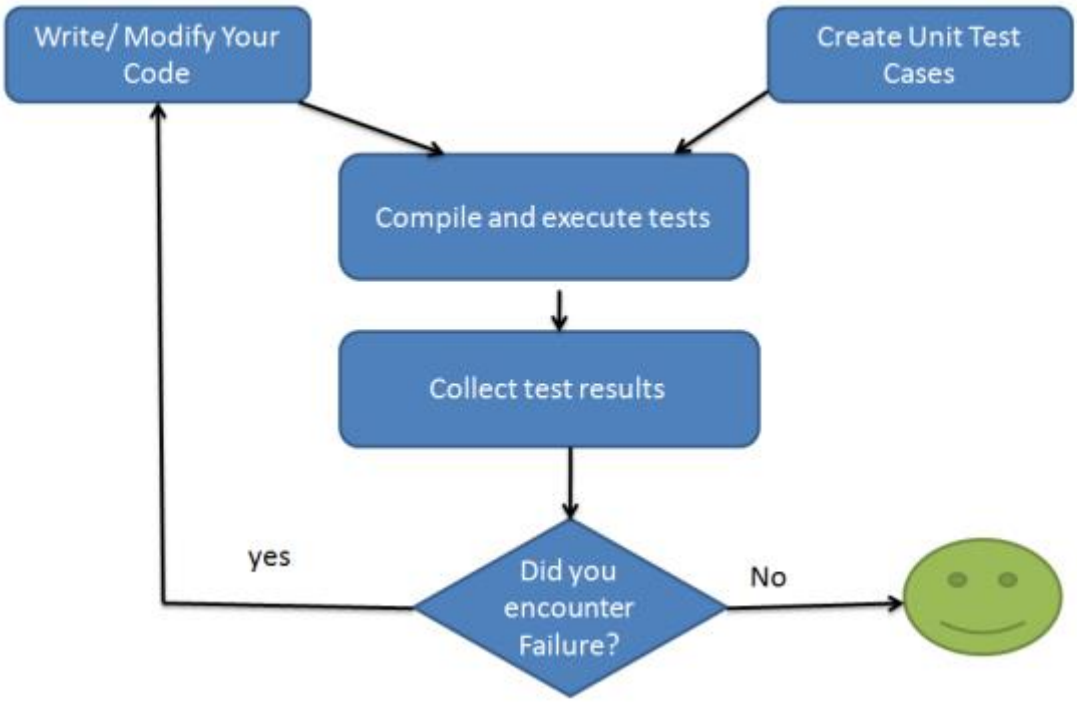
Overview of Software Testing Levels

Unit Testing

The smallest independent and testable part of the source code is referred to as a unit. It is the first step in software testing environment and is generally conducted by the developers or their team

mates. This form of testing is rarely performed by software testers. In order to perform integration testing it is important to first complete the unit testing for all the units. In order to perform unit testing it is important to have well defined unit test plan and unit test cases.

There are several benefits of unit testing. First of all, you get the confidence for going ahead with the integration testing only when you are sure that all units are working correctly. When you start unit testing in parallel to development it may look like a slow process as many defects are uncovered during this stage and several changes are made to the code. However, with time the code is refined and number of defects begins to reduce. So, the foundation of the software is strong and in the later stages the software development is carried out at a much faster pace thereby saving a lot of time.



Unit Testing

If unit testing is carried out properly then it would also result in a lot of cost saving as the cost of fixing a defect in the final stages of software development are much higher than fixing them in the initial stages.

Unit testing is carried on the smallest testable component of the project so the number of test cases and test data are less, and it is not always possible to check all the scenarios for functional and information flow of software application. So, there are many test cases that can be tested only after the unit has been merged with other units to form a bigger component.

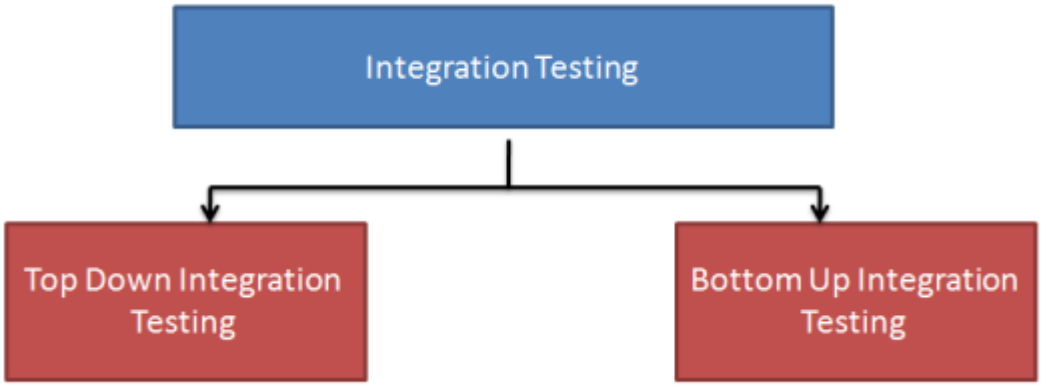
Integration Testing

Once the unit testing phase is over, it is time to move on to integration testing. During integration testing the tester checks how one or more units interact with each other and produce output for various scenarios. This form of testing is carried out a software testing engineer.

In this form of testing a lot of defects related to functional, requirement and performance levels are uncovered. Unit testing confirms that various units are able to perform as per the requirement individually but integration testing confirms whether these independent units are able to perform as per expectations when integrated together. Integration testing can be broadly classified into:

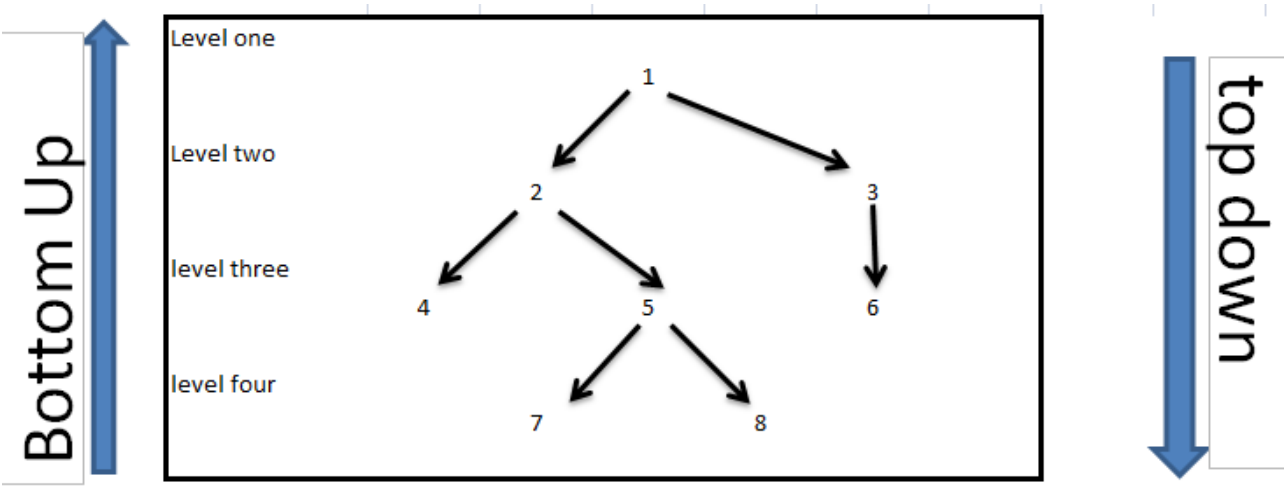
1. Big bang
2. Top down and
3. Bottom up approach

As the name suggests, big bang form of testing all the modules are combined to form a complete system and then tested for bugs.



Integration Testing

Top down is systematic approach where the top level modules are first tested and then one by one the sub modules are added and tested. The Bottom up approach is just the opposite of top down. In this case the lower most modules are tested first and step by step the higher level modules are added and tested. Generally the bottom up approach is followed first in software testing followed by top-down testing.

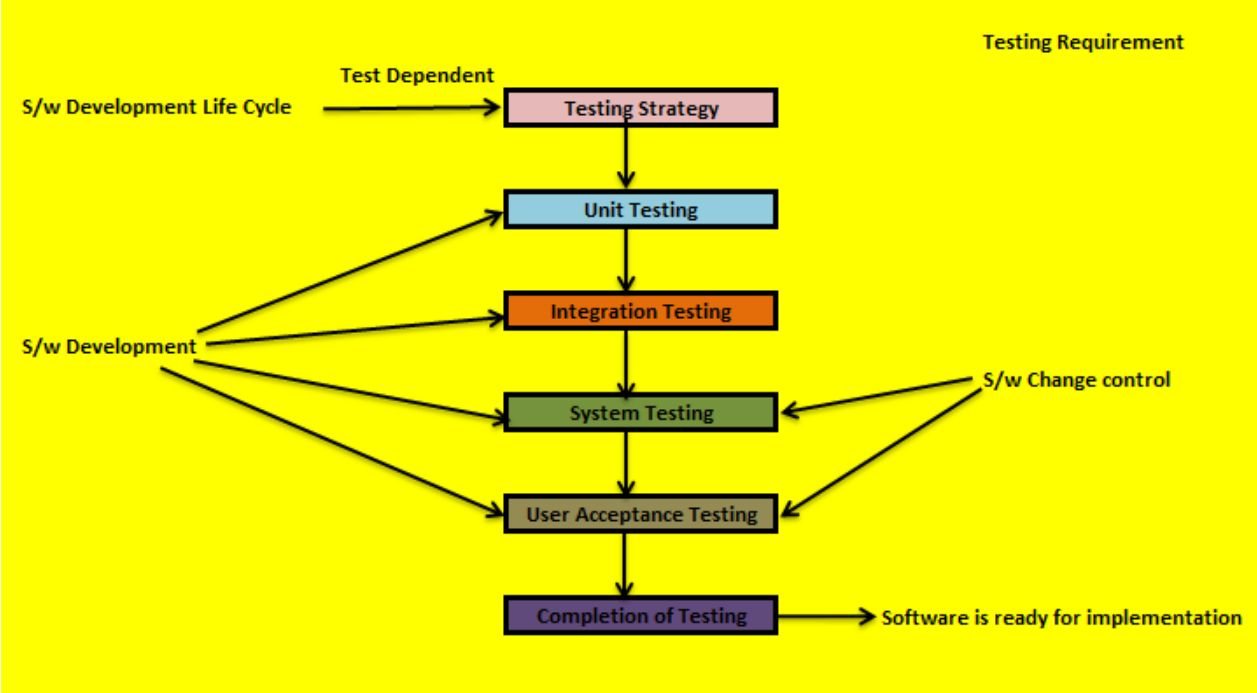


Top down and Bottom up Integration Testing

In Top down approach integration testing stubs can be used as handles in case of modules or sub programs that are not available or not ready. These are dummy modules used at low levels. In similar way in case of bottom up approach is a main program is not available then calling program referred to as driver can be used as a replacement to complete testing process.

System Testing

Once the integration testing phase gets successfully completed it is time to move on to system testing where the system as a whole with all components well integrated is ready for further testing. This is where the software is not only tested for performance but also for adherence to quality standards. As the system is tested as a whole to see if it is in compliance with the functional and technical specifications and the quality standards defined by the organization, it is important that this form of testing is carried out by a highly skilled testing team. For this form of testing it is very important to create a scenario similar to the real time scenario where the system will be deployed.



System Testing

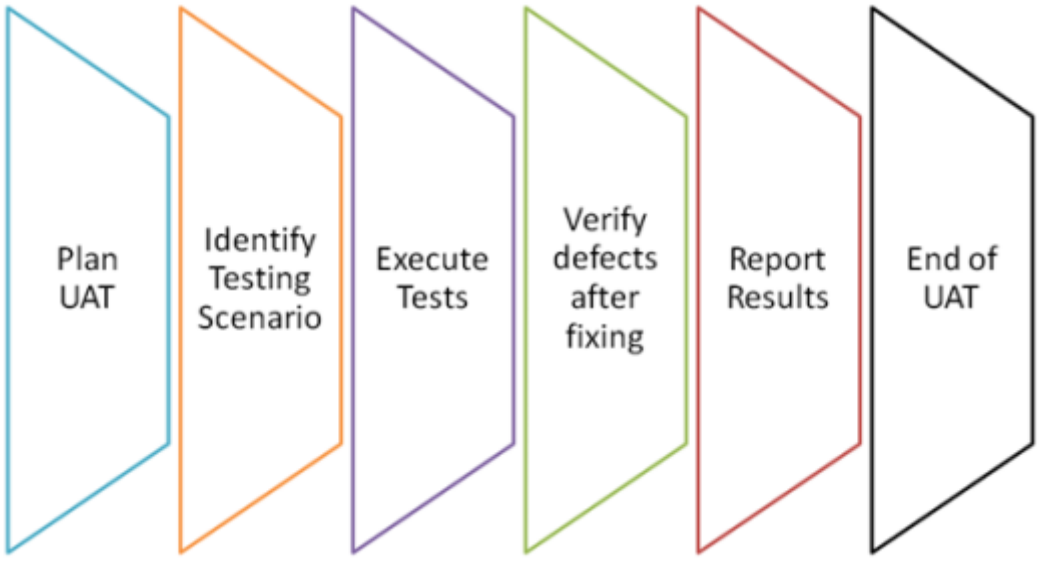
System testing is purely black box testing. The system is checked as per the requirement specifications. The testing is carried out from the user's point of view. This type of testing is carried out to check the behavior of the application, software design and expectation of the end user. System testing validates and verifies both Application architecture and business requirements of the client.

Acceptance Testing

Once the system has been thoroughly tested via unit, integration and system testing it is time for the quality assurance team to come and have a look at the system and test it for quality with the help of predefined test scenarios and test cases. The software is tested for accuracy. The acceptance testing looks at the system from various angles: right from cosmetic looks to internal functioning. This form of test is very crucial because there are legal as well as contractual requirements associated with the software for it to be accepted by the client. Acceptance testing can be of following types:

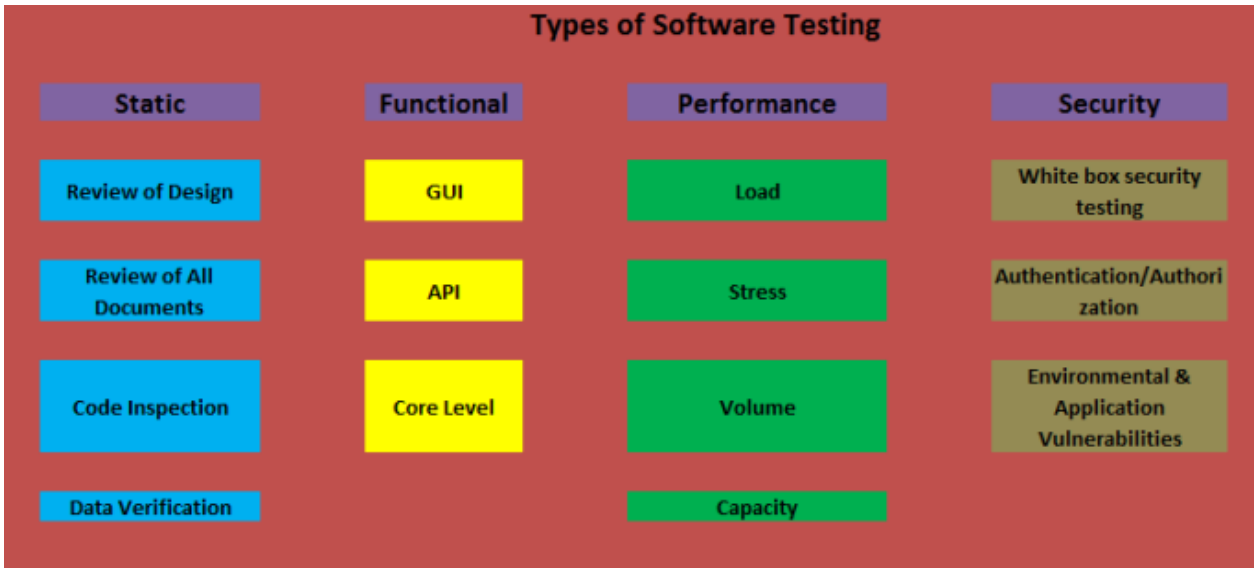
1. User Acceptance testing: This form of testing is carried out by the actual user before the software is accepted. It can be performed at the user's site or in the software organization where the software was developed.
2. Operation Acceptance testing: this form of testing is done to ensure that all the processes and procedures are in place so that the system can be used easily and also be maintained easily.

3. Contract and regulation acceptance testing: is carried out to ensure that the software is in line with all the necessary government, legal and safety standards.
4. Alpha testing: is carried to ensure that the product is of good quality and also to prepare the system for beta testing. This form of testing is performed towards the end of software development where the system can be tested as a whole. This can be a long process as the testers are looking into quality as well as engineering aspects of the developed software. This form of testing is carried out by test engineers and other team members and the product is tested from the point of view of a customer.
5. Beta testing: Once the alpha testing is over, beta testing follows in order to improve the quality of the product and see that the product is as per the requirement of the customer. This form of testing is done a couple of days or weeks before the launch of the product. Beta testing can take a few days but may not take as much time as alpha testing as chances of defect detection are pretty low during this time. It is carried in the real world scenario with the people who are actually going to use the product.



Acceptance Testing

8. Software Testing Types



Software Testing Types

Functional Testing

Functional testing is a kind of black box testing where test cases are prepared keeping the specifications in mind. This form of testing is done to check if the system is in compliance with the client's requirements. Basically in case of functional testing the following checks are important:

1. The tester needs to be very clear about the functionality that the application is supposed to perform.
2. In order to test the application it is very important to have the right set of data(valid and invalid inputs)
3. The output of application for the test data provided should be checked as per the functional specification defined.
4. The test cases must cover all possible test scenarios.
5. The actual result for a given input should be recorded and checked against the expected output.

Types of functional testing include:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Regression Testing
5. Acceptance Testing

Non-Functional Testing

In a software system there are many requirements such as performance, security, user interface etc. that are non-functional in nature however it is very important to test these attributes. Software testing carried out to test non-functional attributes of a system is called non-functional testing. The types of non-functional testing methodologies are as follows:

1. Performance Testing
2. Usability Testing
3. Security Testing
4. Portability Testing

Usability Testing

Usability testing is a process in which the testers test the product to check how easy it would be for the user to use the user interface or in other words the software is tested for its user friendliness. It is a form of black box testing. The software is tested for three things:(1) How convenient the software will be for the user? (2) Will the user be able to use it?(3) Will the user be able to learn it?
Usability has following aspects associated with it:

- How easily the userscan adaptto the system the first time they try to use the system.
- How efficient the system is for the user?The efficiency of the system greatly depends on the speed with which the users are able to accomplish their tasks.
- How easy or difficult it is for the users to work on a system when they get to use it after a long period of time.
- Do the users encounter errors? If yes, then how easily they are able to come out of the issue?
- Are you able to provide satisfaction to your users?

Usability testing assures the end user that the software is of good quality and easy to use. This type of testing very essential in order to satisfy the customers and it needs to be planned well. If planned properly, this activity can be highly beneficial and economical. Since the software is used in a random manner in this type of testing, many times it uncovers defects that have escaped the normal testing procedures.

Graphical User Interface Testing

The form of testing carried out to test if the graphical user interface of an application is working in the right manner is called GUI or Graphical User Interface testing. The GUI is not only checked for proper functionality but also for its adherence to the defined quality standards. The following are some of the most important things that need to be checked during GUI testing:

- Layout
- Colors

- Fonts
- Font size
- Labels
- Test box functions
- How the text is formatted
- Captions
- Buttons
- Lists
- Icons
- Links
- Content
- Short cut keys
- Hour glass while the system is processing data

This form of testing can be manual or automatic whatever is convenient for the tester. The GUI testing is sometimes conducted by third party organizations and not by the development or testing team of the company.

Non-Graphical User Interface Testing

Graphical user interface is the look and feel of an application. Testing anything other than the look and feel of an application such as command lines interfaces, batch processes and various events which trigger certain use cases in a software application come under non graphical user interface testing.

Portability Testing

Portability testing is carried out to test how the change of environment changes the performance of the software. For instance, how the software works on different operating systems or if it is a web-based application, it would be checked for performance on different web browsers. This form of testing is important if the customer intends to use the software applicationfor more than one platform.

This form of testing is a subset of system testing. Here the software may be installed in more than one environment or its executables may be created and run on different platforms. In order to carry out portability testing without any issues it is important to keep all portability requirements in mind while designing and developing the software. This type of testing is performed after integration testing. It is important to have the right testing environment to carry out portability testing.

Security, Authentication And Authorization Testing

A software application is tested for any kind of security flaws in security testing. Authentication and authorization are considered to be two very important aspects of software testing. It is crucial

to carry out this form of testing to ensure that the software is secure and capable of storing confidential customer information when necessary.

Authentication is a process where the person trying to access the software is asked for a username and a password. A wrong combination of the two parameters indicate that the person is not what he is claiming to be and is not allowed to go any further. This is an authentication check which ensures that the login credentials of an authorized software application use are checked.

Authorization on the other hand is all about privileges to use certain restricted features of a software application. Just because you have cleared the authentication process does not mean you have been authorized to access all the data, features and use cases on the software application. For example, in case of social media sites once you feed in the right username and password, you can only access the details of your account and not any other account besides that.

Besides authentication and authorization, security testing also deals with confidentiality, integrity, availability, non-repudiation, software data security, SQL insertion flaws, cross-site scripting attacks and other security related concern which is a complete different subject matter expertise by itself.

Data And Database Integrity Testing

In data integrity testing we need to check if the data stored in the database is accurate and produces results as per the expectations. Some checks are very important for data integrity testing such as whether it is possible to retrieve blank information from the database or not, is data validated properly before getting saved in data base, can the data stored in database be updated, are you able to run tests for all kind of data files etc. In other words data integrity testing is carried out to ensure that the data is accurate and consistent over its entire life cycle.

Database integrity on the other hand deals with testing of all the necessary methods and processes that are important for accessing the database and managing the data within. It is also about how all the access methods functions and data are processed. The data should not get corrupted and should not get deleted, updated or created unknowingly, and there should be test cases to make these validations.

Fault Tolerance And Failover Testing

Fault tolerance testing is carried out to check how an application would react if any fault occurs in real time scenario. In this type of testing various scenarios are considered such as connection error with network, connection problem with application server, not being able to connect to a database etc.

Failover testing on the other hand is carried out to test the ability of software to recover from unexpected severe problems such as hardware failure or a crash. This form of testing explains how well the system will restore to its original state in case of any kind of failure.

Configuration and Deployment Testing

Configuration testing is carried out to verify how a system performs for various types of system configurations. There are several reasons for carrying out configuration testing. It can help you determinethe reaction time of the system in the test environment and the optimal configuration required for the system to perform as expected. Configuration testing also plays a very important role in scenarios where there is a need to migrate a software from one platform to the other. It helps in verifying whether the system is compatible with the new environment as required by the hardware specifications.

Once the system has been deployed, it is important to conduct certain checks to ensure that it is going to work properly. This includes checking the system for its performance issues such as low speed or chances of crashes, how the application is working in real time scenario, is the software employed correctly in the correct folder etc. All this is part of Deployment testing.

Regression Testing

Once a defect is detected in the system it is immediately sent for fixing. However, once the defect is fixed it is important to carry out intense testing in order to check that changes made in the code has not affected any other area of the system. Regression testing is carried out to ensure that bug fixing has not caused any functionality or business logic violation. Regression testing helps in minimizing gaps in testing process. It ensures that the application has no defects before it is sent for next testing phase.

Performance Testing

Subjects such as network delay, data rendering, database transaction processing, load balancing between servers are generally uncovered during performance testing. In other words, rather than finding defects in the actual software, performance testing focuses on testing performance issues. It is important to conduct performance testing in any software for which it is important to have stability, scalability and speed which means good response time and data rendering.

Load Testing And Performance Benchmarking

The process of load testing is also very interesting; here the behavior of the software is tested under maximum load. Load testing is carried out to check how the software performs or behaves under peak load conditions. There are several tools available for load testing such as JMeter, Load Runner, Silk Performer and so on. These tools are used for automated load testing of the software where several virtual users can be created and then a script is executed to check how the software behaves when multiple users try to access the system concurrently. The tester can control the number of virtual users and see how the system behaves.

Stress Testing

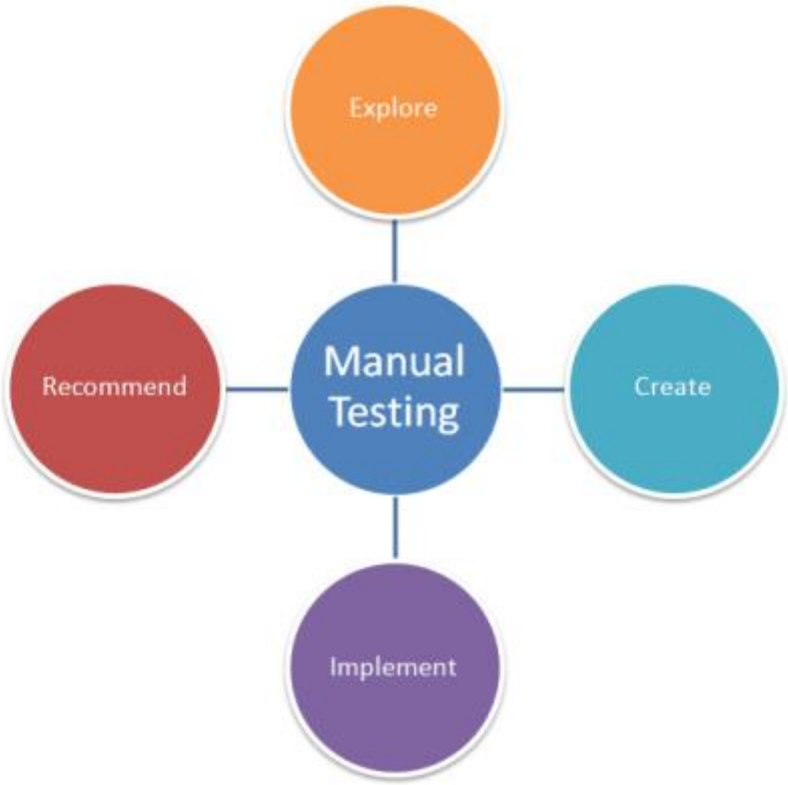
In stress testing the software is intentionally subjected to abnormal conditions and then its performance is monitored. In stress testing the software is tested by applying load that is much beyond the acceptable limit and the resources are withdrawn to find out the point at which it would fail to perform. There are many ways of creating abnormal conditions for stress testing, the database can be turned off and on, and network ports can be shut or restarted randomly. The most common way of performing stress testing is by starting processes that would consume a lot of resources.

	Unit Testing	Integration Testing	System Testing	Acceptance Testing
Functional testing	yes	yes	yes	yes
Configuration testing	no	no	yes	yes
User Interface Testing	no	no	yes	yes
Usability Testing	no	no	yes	yes
Installation Testing	no	no	yes	yes
Document Testing	yes	yes	yes	yes
Performance Testing	yes	yes	yes	yes
Security Testing	no	yes	yes	yes
Scalabity Testing	no	yes	yes	no

Sofware Testing Types Summary

9. Manual Software Testing

Manual testing is the oldest and the most thorough way of conducting software testing. Although many organizations have now started incorporating automation testing in testing projects, automation testing can never completely replace manual testing. The process of manual testing is greatly dependent on the skills and dedication of the tester.



Manual Software Testing

Manual Software Testing Overview

In order to perform manual testing the tester has to be absolutely clear about the testing processes. There is a lot expected from the tester. He needs to understand the functionality of the program because the entire testing activity can be carried out in the right manner only if he has understood the requirement properly. The tester will have to create a test environment and then prepare the test cases accordingly. The process of executing the test cases and verifying the results is also done manually.

The tester has to manually record which tests have passed and which have failed create a detailed report for analysis. Manual testing is of great help when incorporated in the initial stages of software testing.

Manual testing can be incorporate for both big and small projects and wherever there are budgeting issues or the company cannot afford to invest in automation testing tools, manual testing proves to be of great help. Manual testing is still sometimes considered to be more reliable than automation testing although it takes longer automated software testing.

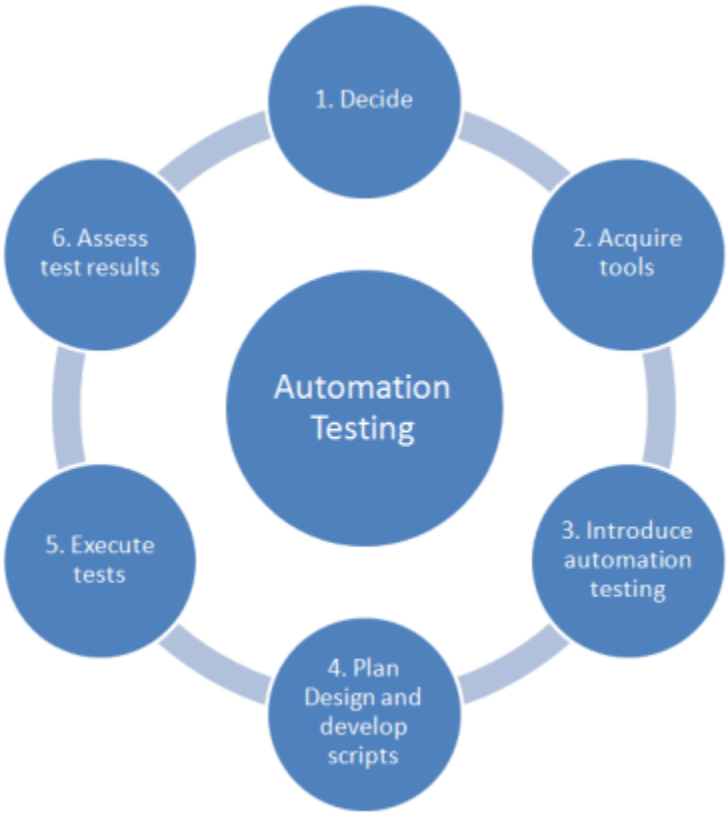
Manual Software Testing Strategy

The strategy for software testing is defined in the test plan. The test plan document defines:

1. The testing environment in which the software will betested
2. The purpose of conducting testing
3. The scope and objective of the testing team
4. How testing activities are schedules
5. The approach for carrying out manual testing
6. Milestones
7. Roles and responsibilities of various members of the test team
8. Tools and Testing Methodology Trainings -if any required-
9. What type of tests need to be performed
10. How defects would be tracked

10. Automated Software Testing

Automation testing is a process in which a testing tool which is also another software application is used to test the system. Test scripts are created and executed and then the results of these tests are compared with expected results. It is always wise to automate repetitive but essential testing processes as it saves a lot of time.



Automated Software Testing

Software Test Automation Overview

Automated software testing is carried out with the help of automation testing tools. Automation testing tool is a software application itself with the help of which a tester can write testing scripts and then use this software to test the actual system under testing. Automation tools are used to automate certain sections of manual testing. The biggest advantage of automation testing is that it saves a lot of time and software application can be thoroughly tested in a short span of time. Automation testing tools are available to perform regression, load and stress testing. However, automation testing cannot completely replace manual testing. Beside time, automation testing saves money and effort and helps in improving the accuracy of software.

All aspects of software cannot be covered via automation testing. However, automation testing is of great help in testing GUI aspects, database connectivity etc. It is best to go for automation testing when you are working on large and complex projects where there is a necessity to test certain areas frequently. Automation testing is also good for testing those applications which will be eventually used by several concurrent users. This form of testing helps in saving a lot of time so the team gets more time to focus on the quality of the software.

In order to perform automation testing, you need to know the following details:

1. Which sections of the software require automation testing
2. Which tools will be ideal for this purpose
3. The process of writing test scripts and executing them
4. You should be able to identify potential bug and know how to create result reports.

It is good to include automation testing to improve the quality of the software but automation testing is not the only way of achieving quality and in no way it should be considered a replacement for inspections and walkthrough procedures. Many organizations still consider that automation testing is the last way of detecting defects that may have gone unnoticed.

Software Test Automation Strategy

There is no doubt that automation testing makes life a lot more easier but just having the right tools is not sufficient to achieve success in automation testing. It is very important to develop a strategy for automation testing that would clearly define which sections of the software require testing, how this task would be carried out, how and where the scripts will be maintained, how the project would benefit from this activity and how much money would be saved in this process. The more specific you are in defining the strategy; the more successful you will be in achieving your testing goals.

While defining the strategy it is important to break down your aim into smaller goals and check if you are able to achieve what you require with the help of an automation test tool. If the project is too big and you try to test complex areas in the first go then there are chances of making mistakes. So, it is wise to start small initially and then gradually grow.

Automation testing can be carried out in unit testing, integration and system level testing. It is always better to uncover maximum defects in early stages of software development hence it is better to plan automation testing as early as it is possible.

Software Test Automation And Its Return Of Investment (ROI)

ROI for Software Test Automation is calculated to know how the company would benefit by incorporating this methodology in its testing processes. It gives a fair idea about the cost saving, scope of improvement in efficiency and software quality.

ROI = (Gain- invested amount)/invested amount
The most commonly used methods for calculation of ROI for automation testing are:

- Simple ROI calculation to know how the company would benefit from cost saving. This calculation is of great importance when a company wants to know how it would benefit on monetary basis by incorporating automation testing. The cost of investment would include the amount spent on acquiring the license, hardware, and training for staff, development of scripts, their maintenance and analysis.
- Efficiency ROI calculation tells about the benefits in terms of increased efficiency. Unlike simple ROI, this calculation only looks at time investment gains. When a company already has the automation testing tools and has been using it for quite some time, there is no need for it to know about Simple ROI calculations because it is not going to make any fresh monetary investment in this case.
- Risk Reduction ROI calculation indicates reduction in risk and scope of improvement of quality. Automation testing saves time and provides team with more time to carry out analysis and carry out ad hoc and exploratory testing this leads to thorough coverage thereby reducing the chances of failure.

Test Cases to Automate

It is difficult and not wise to automate all testing. So, one must clearly determine which test cases should be automated. You must first look at those sections of the software that require repeated testing. Test cases that have to be executed repeatedly and require a lot of data for execution should be automated. Besides, repetitive tests, you should also focus on automating:

1. Test cases that require multiple data sets for execution and are difficult to conduct manually.
2. Load and stress tests that are difficult to conduct manually.
3. Tests that are executed to check the performance of software on multiple platforms.
4. GUI testing.

Test Cases Not To Automate

There are certain test cases that are not preferred to be automated. Such as:

1. Test cases that would be executed only once or quite infrequently.
2. Ad hoc or exploratory testing cannot be conducted with the help of automation testing.
3. Test cases that require only manual execution or a human opinion.

How Do We Want Automat?

Automation of a test case begins with defining what you want to test. Always go for those test cases that are independent and stable. It is always better to execute smaller independent test cases rather than few big and complex ones. Large scripts are difficult to maintain and can be difficult

to execute if any major changes have been made to the software. If a test script requires a change then it is always easier to make changes to a shorter script than a longer one.

Before automating a test case it is better to execute the test case manually. Since automated test cases are those that one needs to use for repetitive testing therefore before designing such a test case it is better to execute it manually and note down all the steps that you would like to record or put down in your test script. This will help you design all the conditions required for through testing of a function.

Ensure that you have taken care of all validations in your test scripts. If you are planning to test database transactions, then you need to make sure that database has the necessary data required for testing.

Before executing the test case it is important to understand how the automated software tool has been designed. This is important because once you start testing you don't want to get interrupted by unknown issues.

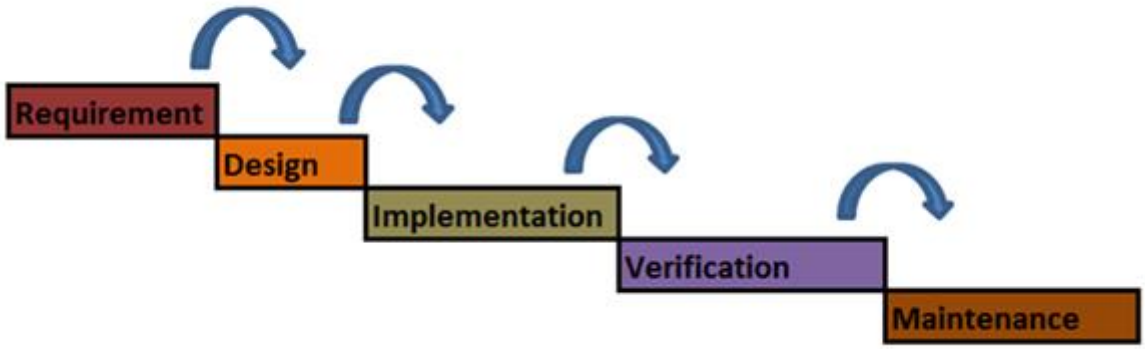
Software Test Automation Tools

Automation tools have a very positive impact on the efficiency and productivity of software. It is important to use software test automation tools in order to simplify testing. Certain manual testing processes can be very time consuming. With the help of automation tools you can accomplish more in less time. However, never depend completely on automation tools. You must clearly define what needs to be tested manually and where there is a need to implement automation testing tools. Most well reputed automation testing tools come with their prices, but it is worth deploying them in projects as they are of great help.

11. Waterfall Software Engineering Life Cycle

The waterfall model is a linear and sequential model defined for software engineering life cycle. It is a classic and very popular model that distinctly defines various phases and the goals that each phase has to achieve. It is called a waterfall model because just like a waterfall once the course of life cycle has started there is no looking back.The water will flow from the top to bottom and not reverse its course. In similar fashion in this life cycle the course of software development cannot be reversed.

In waterfall model the approach is very strict and well defined but there is not much of scope for revision. Testing is a phase and does not go hand in hand with development. So, it is not possible to detect defects early in development stage.



Waterfall Software Engineering Life Cycle

The sequences of phases in waterfall model are as follows:

1. Requirement and gathering: The requirement of the client is captured and all the necessary documentation is done.
2. System Design: based on the requirement gathered the system is designed; this helps in defining in creating the system architecture and also helps in defining the hardware and software requirement of the system.
3. Implementation: Based on the system design the coding begins. The program is divided into small independent units that are later integrated to form the complete system. Once the units are ready they are tested as per the defined specifications.
4. Integration testing: Once the units have been tested they are integrated into a system and then the entire system is tested in one go for defects.
5. Deployment: The system is ready for deployment once the defects uncovered during integration testing have been fixed. After the testing of the integrated system, the defects

are fixed and testing is done again to verify is all defects have been closed. The system is now ready for market release and can be deployed in the customer's environment.

6. Maintenance: If any defect or issue arises after the software has been delivered to the customer then patches or new version of the software is released to deal with the problem.

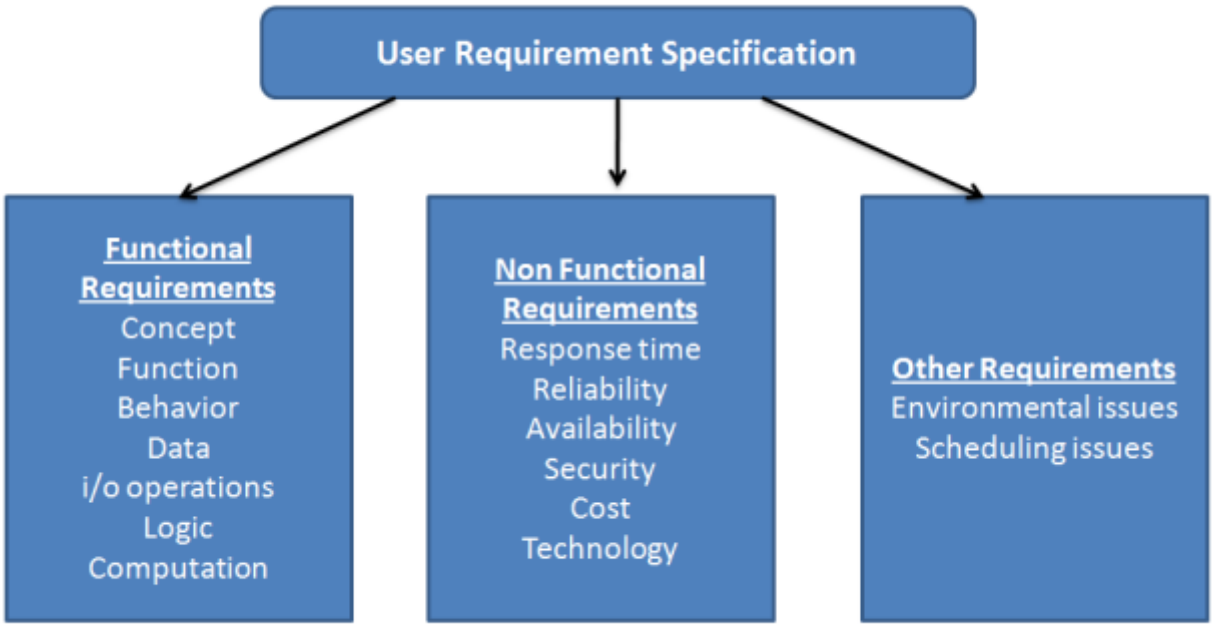
All phases mentioned above are cascaded with each other and one phase can start only after the previous phase has been closed successfully.

Waterfall model can be used in situations where the requirements are well defined and clear, the product is stable and there are enough of resources available for each phase. Waterfall model should be practically implemented only in small and easily manageable projects. It should not be used in projects in which the requirements are at moderate to high risk.

Requirements Specification and Analysis

The development of a software application depends only one thing and that is how well the requirements are captured and documented. This process of identifying and then documenting the requirements for software development is called Requirement specification and analysis.

The client, the management team and the technical support team of the company discuss the requirement of the software in complete detail. These requirements are analyzed, discussed several times and then documented till they are clear and complete.



User Requirement Specification

The requirements can be documented in the format defined by the company such as description in word docs, flow charts, drawings etc. The outcome of the entire exercise is a detailed document that describes what the customer wants. This is necessary so that one is able to understand the requirement of the client properly. In present day we say that defect is anything that is not in line with the requirement of the customer. If we capture the requirement incorrectly we will start the project on the wrong foot.

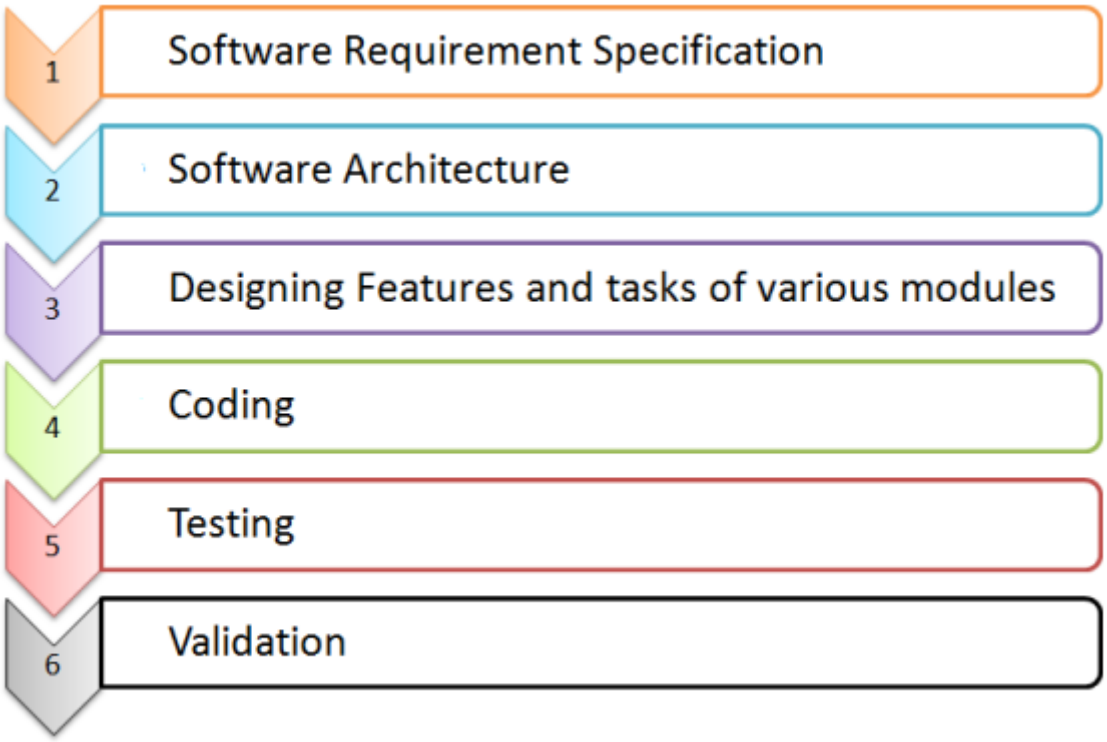
User Requirement Specification

This document defines what a user wants and what his expectations from the software are. This document gives a clear picture about the business needs of a client. This document is written by the end users. These requirements are tested in the user acceptance testing. This document is not technical in nature. It talks about the business need of the user and forms the foundation for planning the next steps in capturing the requirements and system design.

Software Requirement Specification

The software requirement specification defines how the user wants the software to function. The purpose of this document is as follows:

- The customer and the supplier agree on this requirement. It describes the functionality of the software in complete detail.
- The software design is prepared on the basis of SRS document.
- Clear cut SRS document simplifies development and testing process.
- It helps in scheduling activities and estimating costs.
- SRS serves as a foundation for validation and verification.

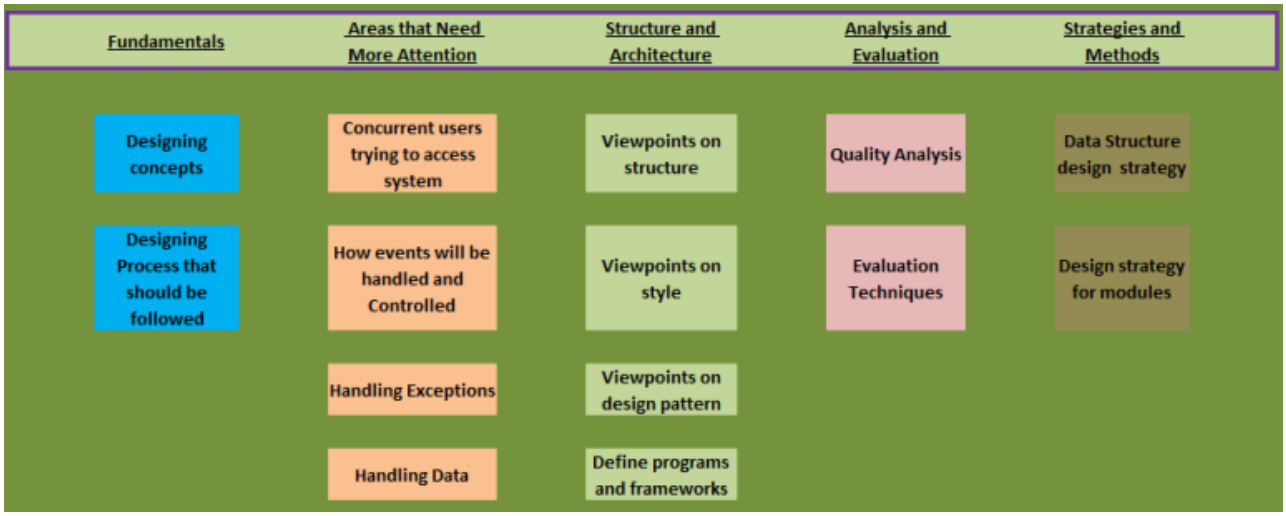


Software Requirement Specification

Software Design

Software Design can be of the following types:

- High Level Software Design
- Detailed Software Level Design



Software Design

High Level Software Design

High level software design provides a clear picture about the database design and the function architecture of the system. High level Software design serves as a guide for the development team. Whenever there is any doubt regarding the design flow, the developers refer to High Level Design (HLD) document. In order to create this document it is important that the Software Requirements Specification (SRS) document should be absolutely clear.

Detailed Software Level Design

Detailed software level design or Low Level Design (LLD) involves breaking the high level design in smaller sections and writing down the logic that would serve as program specification. LLD provides detailed information about the system and it cannot be created till the HLD is ready.

Software Design Processes

Software design Processes can be classified as:

- Top Down Design Approach
- Bottom Up Design Approach

Top Down Design Approach

Top-down approach is a very systematic approach in which first the outline of the system is devised without defining details of any subsystem. Once the outline for the system is ready it is

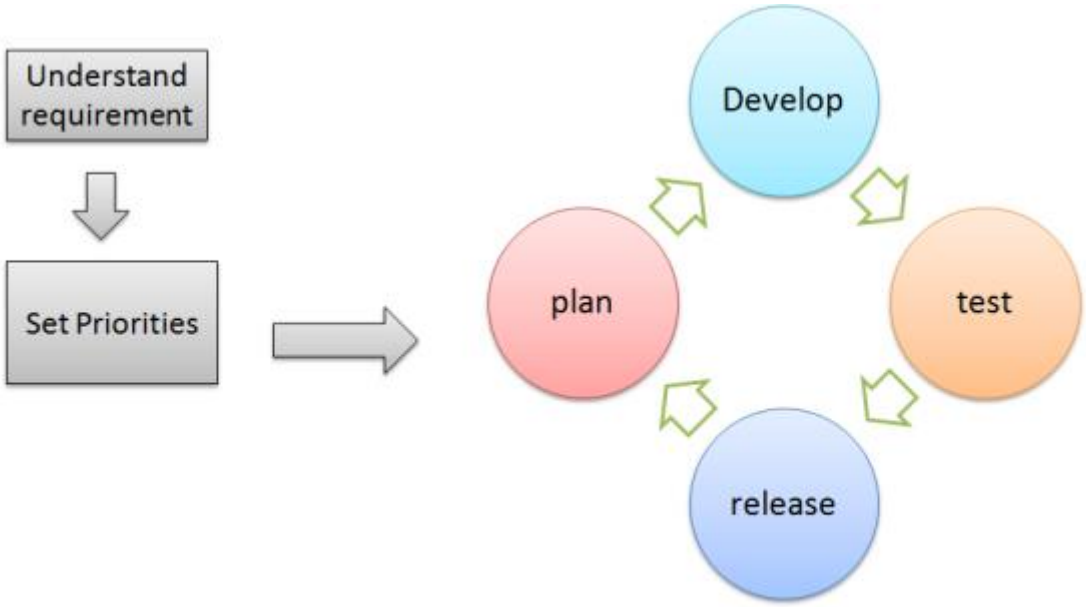
time to move down step by step to every subsystem. Every sub system is defined in complete detail till the specification reaches the base elements.

Bottom Up Design Approach

Bottom up design approach is exactly the opposite of top down approach. In this case the first step involves defining the base elements in complete detail and then linking these elements to form various sub systems which are further linked together till the time the complete system is formed. This approach can be compared to planting a tree where at the beginning all that we have in hand in a seed but as we start working on it a tree is formed.

12. Agile Software Engineering Life Cycle

The agile methodology believes that every project must be handled in a different manner. Requirements of a project vary from client to client and therefore it is not wise to stick to just one method of software development. In software engineering life cycle all projects are divided into small time frames, with each time frame focusing on delivery of certain sections for release.



Agile Software Engineering Life Cycle

This process combines iterative and incremental approach that helps in fast development of the project. The project is broken into incremental builds and then every build undergoes iterations that can last for 2-3 weeks. In every iteration cross functional teams work together on various aspects of the project right from requirement analysis to acceptance testing. At the end of each iteration the outcome is shown to the customer.

Overview Of Agile Software Engineering Life Cycle

Agile software engineering life cycle is different from other software development approach because it is adaptive in nature as compared to others that are predictive. Predictive planning requires in depth planning which consumes lot of time and effort and even a small change in the requirement after the development has started affects the development process. Predictive methods are dependent completely on requirement analysis and planning at the beginning of the project.

Agile methodology does not require detailed planning, development begins keeping the features and characteristics of the software in mind and the team changes the course of development dynamically whenever a change in requirement is requested. This approach focuses more on customer interaction and less on documentation so that the development team is sure that it is on the right path.

Although agile software development follows a very realistic approach it is not ideal for complex projects and there is always a risk of sustainability, maintainability and extensibility. This process has minimum resource requirements and ideal for projects where requirement undergo changes frequently. Since the documentation is less and the focus is completely on customer interaction, the entire project depends on how well the customer is able to communicate his requirements. Agile methodology enables concurrent development; the rules are nominal that can be easily employed.

Continuous Software Integration and Continue Software Testing

In traditional way of software development, the developers work on individual pieces of code for several days and once the individual units are completed they move on to integration of units. Since in every iterationthis methodology believes in making the project focus on development of high quality code, continuous integration should be followed. The concept of continuous integration requires a lot of discipline and theories may vary from company to company. For example some companies believe that at the end of the day a developer must ensure that nothing is left unintegrated. In such a scenario the developer needs to plan all his tasks properly. Although, this may seem like a difficult task, the biggest benefit of this approach is that the customer can walk in anytime and see how the product is being developed and can give his feedback on what is presented to him.

Testing and Role Of Testing In Agile Software Engineering Life Cycle

Traditional ways of software development does not utilize a tester to his complete potential. As a matter of fact testers started working only after the functional requirement has been developed completely. In an agile environment the tester is a very important member of the team and he is involved in every phase of every iteration, be it planning or requirement analysis. In this methodology testing is as important as development and the product is subjected to continuous testing. The tester is working continuously in agile methodology and so, by the end of the project the number of defects in the system are very less in number because a majority of them have been uncovered in initial phases of software development.

13. Software Project Management

Software Project management includes all the necessary information and tools that are required to manage the process of software development projects. In order to execute a software development project plan properly, the manager drafts a plan that describes how the development will be carried out. The plan also focuses on how the quality standards will be maintained during the development process. The manager also defines how the development team will be organized, how risk management will be carried out.

Project Planning

Software Project Planning is required in order to control and monitor the complex processes involved in software development. Project planning is done to ensure that the project is completed well in time, the end product is of good quality and the project gets completed well in time. In order to go ahead with project planning it is important to take into account the project complexity, its size and level of structural uncertainty. A software project plan comprises of:

- Scope: defines the problem and what needs to be done to resolve it.
- Estimation: defines the effort and time that will go in completing the project.
- Risk: Defines the obstruction that the team may face during software development and how that can be tackled.
- Schedule: Allocation of resources so that the project can finish in time.
- Control Strategy: defines how to go about quality control.

Project Staffing

In order to conduct project staffing it is important to define various roles, skills required for each role, how staffing will be scheduled and how staffing will be done for each role. For every role it is necessary to define the skills and competency requirements, the experience level, availability and salary expectations.

Project Coordination and Alignment of Stakeholders

It is the responsibility of the IT project coordinator to align the internal team members with the external stakeholders. This is done by coordinating various project phases and schedules, tracking progress and making arrangements for order supplies and support services. The project manager monitors the project coordination activities.

14. Software Testing Life Cycle And Software Testing Operations

Software testing life cycle has the following phases:

1. Requirement study: where the requirement of the client is understood and functional documentation is done.
2. Test plan: Considering the requirement of the project, testing activities are planned and test plan document is prepared.
3. Test Case Design: Testing scenarios are identified and test cases are designed accordingly.
4. Requirement and test cases are mapped and requirement traceability matrix is prepared.
5. Execution of test cases: tests should be executed and bugs should be reported.
6. Once the defects are fixed, retesting and regression testing must be carried out.
7. Test reports should be prepared.
8. Testing activity is completed.

Requirements Study

In this process:

- Requirement traceability matrix is prepared.
- Module wise priorities and approach is set.
- Area that require automation testing and the ones that require manual testing are identified.

Test Case Design And Development

This process involves:

- Preparation of test cases and test scripts(if automated testing is involved)
- Preparation of data for testing

Test Execution

Test Execution consists of:

- Execution of test cases
- Use a bug tracking tool to log bugs
- Document test results
- Use the RTM to map defects to the requirement
- Track the bug's status

Test Closure

Test Closure consists of:

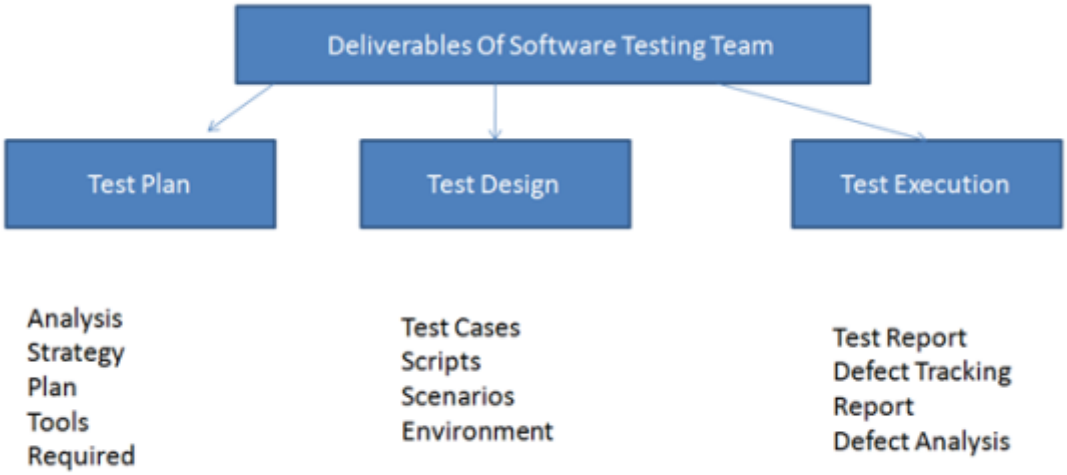
- Test metrics is prepared considering test coverage, cost, time, quality etc.
- Preparation of test closure report.
- Defect distribution by type and severity is calculated.
- Casual analysis and resolution(CAR) is prepared.

Test Process Analysis

Test process analysis is required before getting started with software testing. Documents that provide information to start off with testing are called test basis documents. These include SRS, system design, architecture, interface etc. With the help of these documents the testers understand the system and look for test conditions or possibilities that will be part of the test case. Once the test conditions have been identified they are prioritized and test cases are created.

15. Deliverables Of Software Testing Team

Test Deliverables are documents that are given to the stakeholders when the software is being developed. In this section we will discuss the following test deliverables:



Deliverables Of Software Testing Team

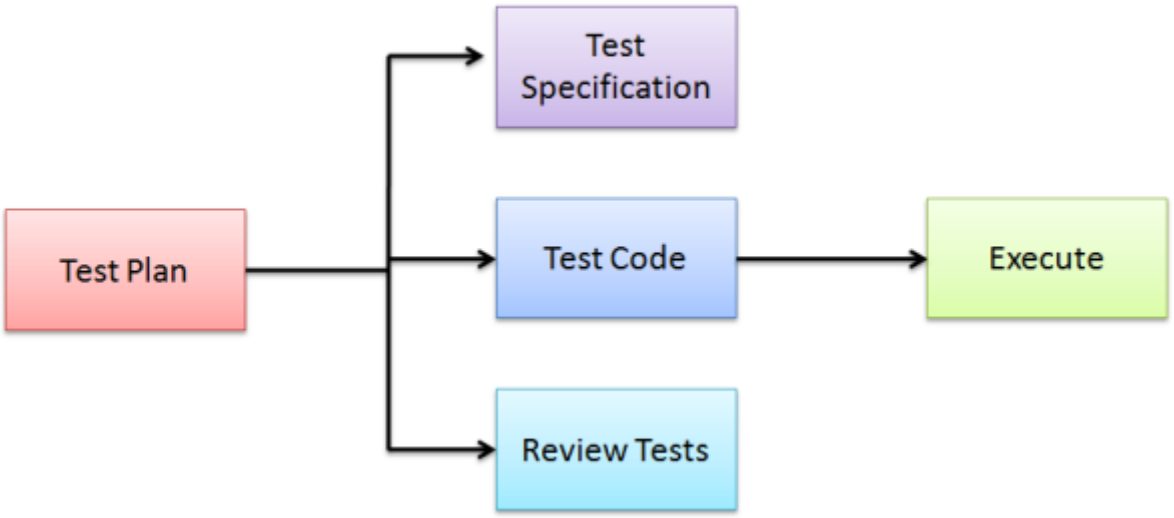
- Software Testing Strategy
- Software Testing Plan
- Software Test Scenarios and Test Cases
- Software Test Metrics
- Product Metrics
- Process Metrics
- Software Test Documentations
- Software Testing Reports
- Daily Test Status Reports
- Incident Reports
- Final (Test Project Closure) Test Status Report

Software Testing Strategy

A software testing strategy is a roadmap for testing activities in a project. Testing can be a very difficult if it is not broken down into smaller well defined processes. A test strategy helps in defining the testing process and on the basis of which the testing team evaluates the time, effort and resources required for the entire process. A good strategy is the one that allows good planning, management and tracking of processes.

Software Testing Plan

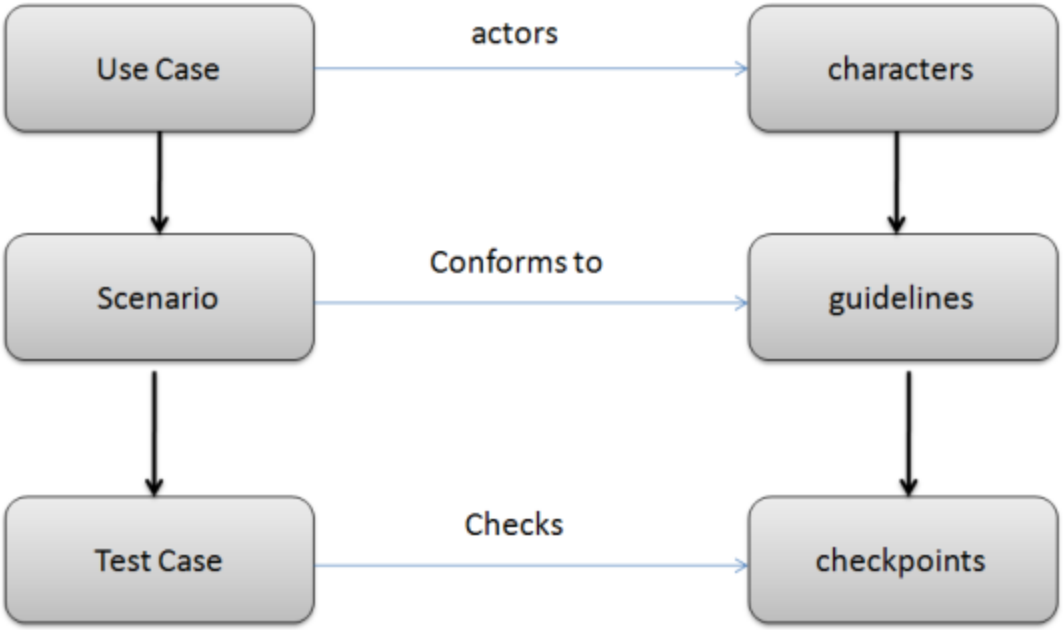
A test plan defines the overall strategy for testing software. It explains the required testing environment and what level of testing will be conducted. The test plan imparts information on how test results will be analyzed and which metrics will be used for the purpose. Exit criteria for each phase are defined and estimate of test effort and cost are mentioned in the plan.



Software Testing Plan

Software Test Scenarios and Test Cases

Test Scenario is a hypothetical case made to help the tester test the software in a well-defined manner. Test cases on the other hand are very simple and straight forward that explain what needs to be done. They are generally one line statements and you need to feed in an input and check the expected output for every input. Test scenarios can be complex and compelling like a story. It is a situation given to a tester to evaluate the software.



Software Test Scenarios and Test Cases

The concept of test scenarios came into existence in the year 2003 after it was observed that maintenance of test cases with step by step description, input data and expected results was a very complicated process. Then the industry has come up with something that was easy to use.

Test cases on the other hand are very direct. They tell what the tester should do and how the software needs to be tested.

Software Test Scenarios and Test Cases

Software Test Metrics

As the name suggests software metrics is nothing but the standards of measurement defined to evaluate software. Any unit used to measure any attribute of software is called a metrics. Software test metrics is used to measure the quality of software. These metrics give a fair idea

about the readiness of a system while it is still under development. Now you may ask why do we need a software test metrics? Well, the answer is very simple- we can only improve those things that we can measure, because anything that is measurable is controllable. Decisions of every phase of software development are taken only after referring to the software test metrics of the previous phase. Software test metrics can be of two types:

1. Direct measure/Base Metrics: this data is raw and is gathered by a test analyst. Base metrics provides the information regarding the status of the project such as how many test cases have been created, how many have been executed and so on.
2. Indirect measure/Calculated Metrics: Raw data collected from base metrics when converted to more useful information that gives accurate information about the project such as % of project completion, % of test coverage and son on are referred to as calculated metrics.

Product Metrics

Product metrics plays a very important role in defining product quality improvement initiatives. Constant evaluation of the product helps the team decide if there is a need for any type of correction in any characteristic of the product well in time. Product metrics measures the characteristics of a product:

- Size
- Complexity
- Design aspects
- Performance
- Quality

Process Metrics

Process Metrics measure the processes so that if there is any scope of improvement that can be done. Process metrics can help in saving a lot of time and effort. Every process is designed keeping some target in mind and process metrics helps in assessing how far the team has been successful in achieving it. Process metrics play a very important role in tracking software testing activities. Software testing is a very important aspect of software development and efficiency and effectiveness of a project can improve significantly by monitoring testing activities. From the point of view of software development process metrics can be used to measure:

- Software Development
- Software Maintenance
- Software Testing

Software Test Documentations

Software testing documents that prepared during or before testing to keep a track of all the testing activities are called software test documents. The important software testing documents include:

- Test Plan
- Test Scenario
- Test Case
- Traceability Matrix

Software Testing Reports

In this section we will discuss the following reports:

- Daily Test Status Reports
- Incident Reports
- Final (Test Project Closure) Test Status Report

Daily Test Status Reports

Daily test status report helps in maintaining transparency within the test team and with the project and program stakeholders. At any point of time the project manager can have a look at how many defects are reported on daily basis, how many test cases are run and how much work was done by each tester. A daily status report has information regarding:

1. How many test cases were planned for the day?
2. How many of them were executed?
3. How many defects have benn reported by the testing team?
4. How many of the defects were critical?
5. Location: Where the more detailed test reports can be viewed (Defect and Test Execution Sheet)?

Incident Reports

Incident reports help in coordinating testing and development activities. Incident management involves:

- Comparison of actual and expected results
- What was the cause of deviation in cases where tests failed? – faulty test automation, poor test case, actual failure of software?

Incident report needs to be prepared for actual failure. An incident report should provide the following details:

- Identification
- What is the ID for each report?
- What is the test object?
- Version of Software Application under Test
- Details of test environment - hardware and software
- Name of the developer who needs to assess and resolve the incident

- When was the failure observed?
- Classification
- Status
- Severity
- Priority
- Requirement
- Cause
- Problem Description
- Details of the test case
- Tester's comments
- Developer's comment after change
- Any references

Final (Test Project Closure) Test Status Report

Test project closure report is created when the exit criterion of the project is reached or when the entire testing activity has been completed. This report is created by the test lead and later reviewed by all the stakeholders and clients. This report has all the details of the test cases that have been successfully executed and if there is any outstanding defect that the team plans to resolve post project deployment then that is also mentioned in this report.

16. What Is Software Risk And Software Risk Management?

Risk is an expectation of loss, a potential problem that may or may not occur in the future. It is generally caused due to lack of information, control or time. A possibility of suffering from loss in software development process is called a software risk. Loss can be anything, increase in production cost, development of poor quality software, not being able to complete the project on time. Software risk exists because the future is uncertain and there are many known and unknown things that cannot be incorporated in the project plan. A software risk can be of two types (a) internal risks that are within the control of the project manager and (2) external risks that are beyond the control of project manager. Risk management is carried out to:

1. Identify the risk
2. Reduce the impact of risk
3. Reduce the probability or likelihood of risk
4. Risk monitoring

A project manager has to deal with risks arising from three possible cases:

1. Known knowns are software risks that are actually facts known to the team as well as to the entire project. For example not having enough number of developers can delay the project delivery. Such risks are described and included in the Project Management Plan.
2. Known unknowns are risks that the project team is aware of but it is unknown that such risk exists in the project or not. For example if the communication with the client is not of good level then it is not possible to capture the requirement properly. This is a fact known to the project team however whether the client has communicated all the information properly or not is unknown to the project.
3. Unknown Unknowns are those kind of risks about which the organization has no idea. Such risks are generally related to technology such as working with technologies or tools that you have no idea about because your client wants you to work that way suddenly exposes you to absolutely unknown unknown risks.

Software risk management is all about risk quantification of risk. This includes:

1. Giving a precise description of risk event that can occur in the project
2. Defining risk probability that would explain what are the chances for that risk to occur
3. Defining How much loss a particular risk can cause
4. Defining the liability potential of risk

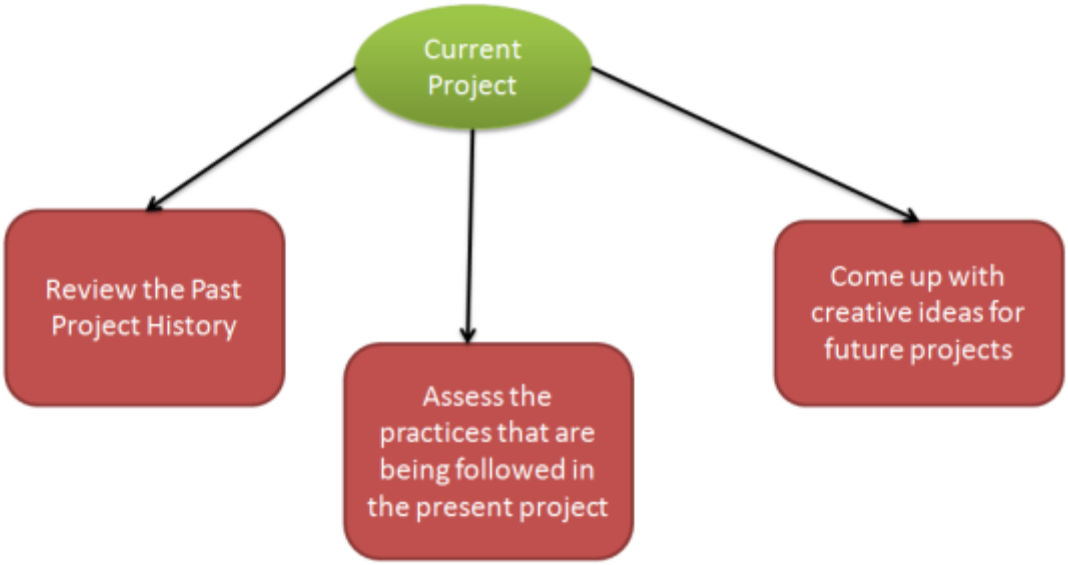
Risk Management comprises of following processes:

1. Software Risk Identification
2. Software Risk Analysis
3. Software Risk Planning
4. Software Risk Monitoring

These Processes are defined below.

Software Risk Identification

In order to identify the risks that your project may be subjected to, it is important to first study the problems faced by previous projects. Study the project plan properly and check for all the possible areas that are vulnerable to some or the other type of risks. The best ways of analyzing a project plan is by converting it to a flowchart and examine all essentialareas. It is important to conduct few brainstorming sessions to identify the known unknowns that can affect the project. Any decision taken related to technical, operational, political, legal, social, internal or external factors should be evaluated properly.



Software Risk Identification

In this phase of Risk management you have to define processes that are important for risk identification. All the details of the risk such as unique Id, date on which it was identified, description and so on should be clearly mentioned.

Software Risk Analysis

Software Risk analysisis a very important aspect of risk management. In this phase the risk is identified and then categorized. After the categorization of risk, the level, likelihood (percentage) and impact of the risk is analyzed. Likelihood is defined in percentage after examining what are the chances of risk to occur due to various technical conditions. These technical conditions can be:

1. Complexity of the technology
2. Technical knowledge possessed by the testing team
3. Conflicts within the team
4. Teams being distributed over a large geographical area
5. Usage of poor quality testing tools

With impact we mean the consequence of a risk in case it happens. It is important to know about the impact because it is necessary to know how a business can get affected:

1. What will be the loss to the customer
2. How would the business suffer
3. Loss of reputation or harm to society
4. Monetary losses
5. Legal actions against the company
6. Cancellation of business license

Level of risk is identified with the help of:

Qualitative Risk Analysis: Here you define risk as:

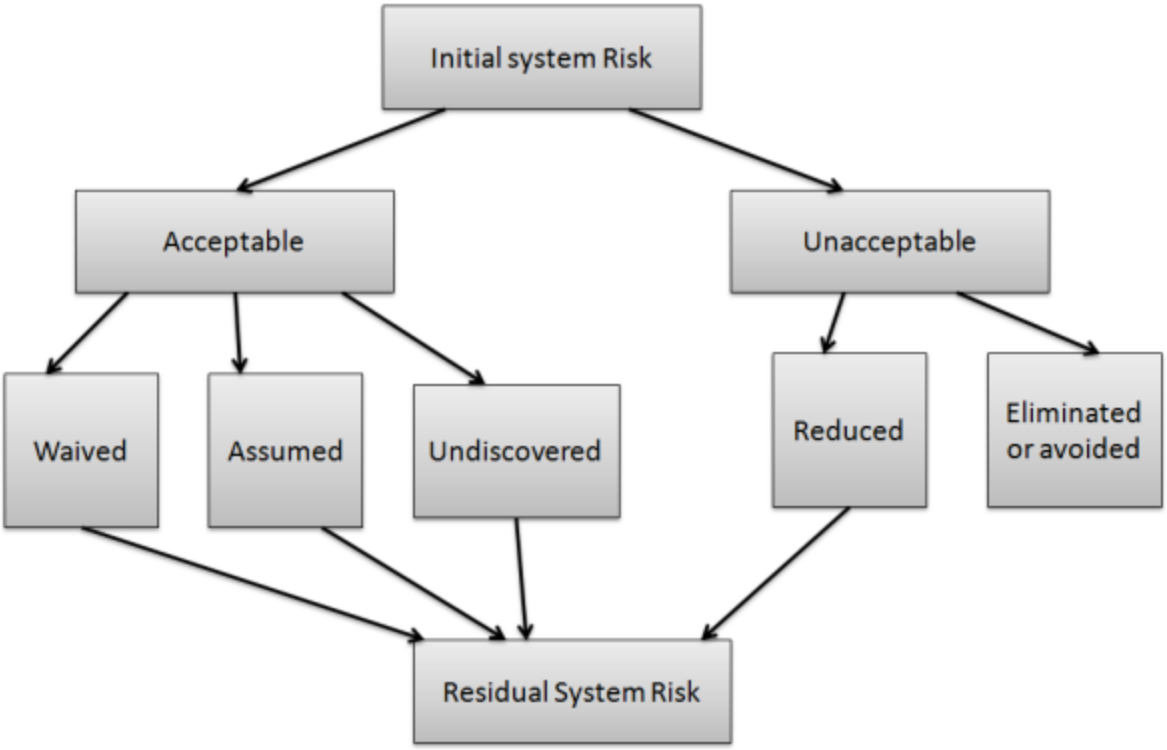
- High
- Low
- Medium

Quantitative Risk Analysis: can be used for software risk analysis but is considered inappropriate because risk level is defined in % which does not give a very clear picture.

Software Risk Planning

Software risk planning is all about:

1. Defining preventive measure that would lower down the likelihood or probability of various risks.
2. Define measures that would reduce the impact in case a risk happens.
3. Constant monitoring of processes to identify risks as early as possible.



Software Risk Planning

Software Risk Monitoring

Software risk monitoring is integrated into project activities and regular checks are conducted on top risks. Software risk monitoring comprises of:

- Tracking of risk plans for any major changes in actual plan, attribute, etc.
- Preparation of status reports for project management.
- Review risks and risks whose impact or likelihood has reached the lowest possible level should be closed.
- Regularly search for new risks

17. Processes to Support Software Testing

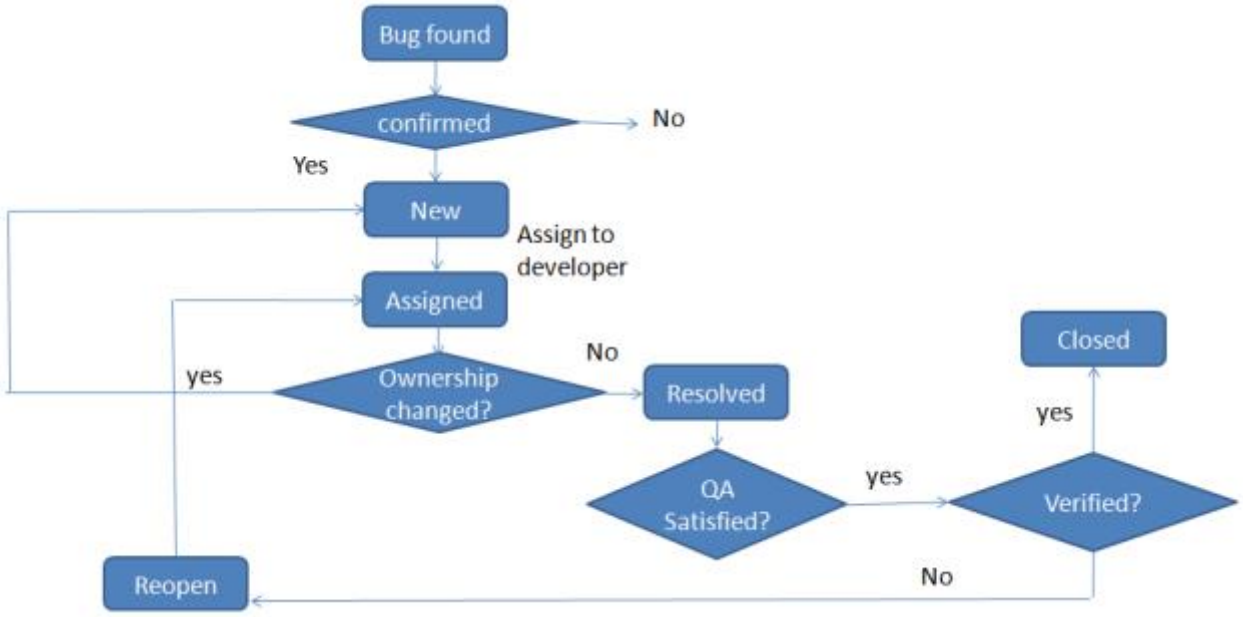
This section focuses on some very important processes that are very essential for keeping a track of how software testing is progressing. In this section you will learn about:

- Defect Management Life Cycle And How it works
- Traceability Matrix
- Software Testing Estimation Techniques
- Configuration Management
- Change Management

What Is Defect Management Life Cycle And How Does It Work?

Software is developed by humans and it is unfair to expect a software of millions lines of code to be defect free. Defects are induced due to many reasons:

1. The requirements are not communicated clearly
2. Programming errors caused during development
3. Complex system design
4. Poor quality or incomplete documentation
5. Difficulty in managing changes in software designsystem.



What Is Defect Management Life Cycle And How Does It Work?

Defect lifecycle is defined to manage defects in the system. It is in fact the journey of a bug in the system right from the time is was found to the time it is closed. The stages of a defect life cycle are given below:

1. New: When a defect is discovered it is given the status 'New'.
2. Opened: When a new found defect goes for review it has a status 'Open'
3. Duplicate: If the defect has already been reported before, then it is called 'Duplicate'
4. Assigned: This status indicates that the bug has been assigned to a developer for fixing.
5. Re-tested: Once the defect is fixed the tester conducts testing to check if the defect has been closed or not.
6. Reopened/closed: if the defect is fixed it is assigned the status 'closed' or else it is 'reopened'.

What Is Traceability Matrix?

Traceability matrix is a tool that is used to connect and trace the requirements of the project (business, application, security related) to the implementation and testing processes. This helps in analyzing how much of the project requirements have been completed.

A traceability matrix is of great use in complex software development projects. With the help of this tool you can trace how things are working in any section of the project. This matrix is created in a worksheet document that comprises of a rows and columns. One set of values are set against the row and other set of values is set against the columns. If there is any kind of relationship between any value of the column and any value of the row then an identification mark is place in the cell where that column and row intersect.

So, if we place the various requirements of the project against the column on the left and various testing processes on the top of row. Then we can easily map which testing processes have been completed for which all requirements. This would give an accurate idea about how much percentage of requirements has got completed.

What Are Software Testing Estimation Techniques?

Different software project belong to different domains so software estimation techniques may vary a bit from project to project. Also it is very difficult to get the estimation right in the first go. However, software estimation techniques are very important for a company. It plays a very important role in the SDLC. Estimation techniques provide an estimate of time that may be required to finish a particular task.

Estimation is done to get a rough idea about how much effort it would take to complete a particular task. The estimation could also mean cost. Many factors such as past experience, assumptions, documents, calculated risks etc. help in determining the estimate for a software development cycle. This estimate helps an organization in bidding for a project. Software estimation is required so that the chances of overshooting the budget while testing or delay in completion of project can be avoided.

1. Delphi software estimation technique is the most widely used techniques for estimation.It provides a way of achieving an agreement within a team without any conflict. Each one from the team provides an anonymous and individual estimate which is assessed by the coordinator. If the assessment is within the acceptable range then a decision can be taken or else the process is repeated again.
2. Work Breakdown Structure is used is big projects where first the project is broken into smaller components in a hierarchy and then the estimation process is carried out to evaluate task scheduling and detailed cost estimation of the project.
3. Three point estimation technique is also used for big projects where the project is first broken into smaller sub sections and the for each sub section first the optimistic estimate is done assuming that nothing will go wrong and all conditions are fine(A). The second approach is of most likely estimate where one assumes that there is a possibility for problems to arise but eventually everything will be fine(B). The third approach is the pessimistic estimate where everything will go wrong(C). So, the estimate formula is defined as: A+(4*B)+C/6
4. Functional point method suggests that: (Total effort Estimate) =(Total number of function points) * (estimate of every functional point). The above mentioned formula clarifies that more weightage is given to every function point. The function points are categorized as complex, medium and simple and then an estimate is done.

What Is Configuration Management?

Configuration management is required to maintain consistency in the performance of a system. Configuration Management involves:

- All configuration items are labeled with unique identifiers
- Identification of all documents that describe configuration item
- Related configuration items should be grouped into baselines
- After every revision it is important to label the configuration items and baselines.

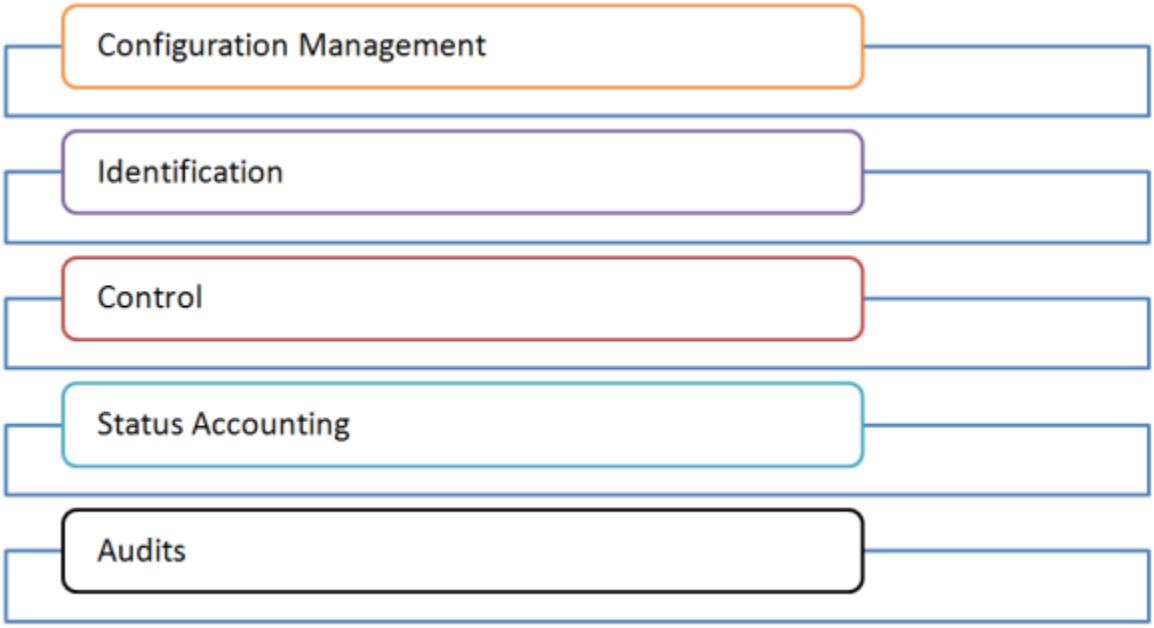
Development of software is a complex process where following issues are very common:

1. Many times developers tend to overwrite each other's modifications
2. If versions are not maintained properly then integration process can be very difficult.
3. Sometimes there are compatibility issues between the versions of the compiler and development tools.
4. Confusion over which test cases belong to current version of software.

All these problems can be handled with the help of configuration management that defines:

- Version management: maintaining the record of various versions of object under test corresponding version of configured item.
- Configuration identification: helps in uniquely defining the system, every revised version of the system and components of each revised version.

- Documentation of incident status and change request: These documents impart information on various incidents and the change requested and their present status.
- Configuration audits: ensure that the details of all components of the software are well documented and all configurations are easy to identify.

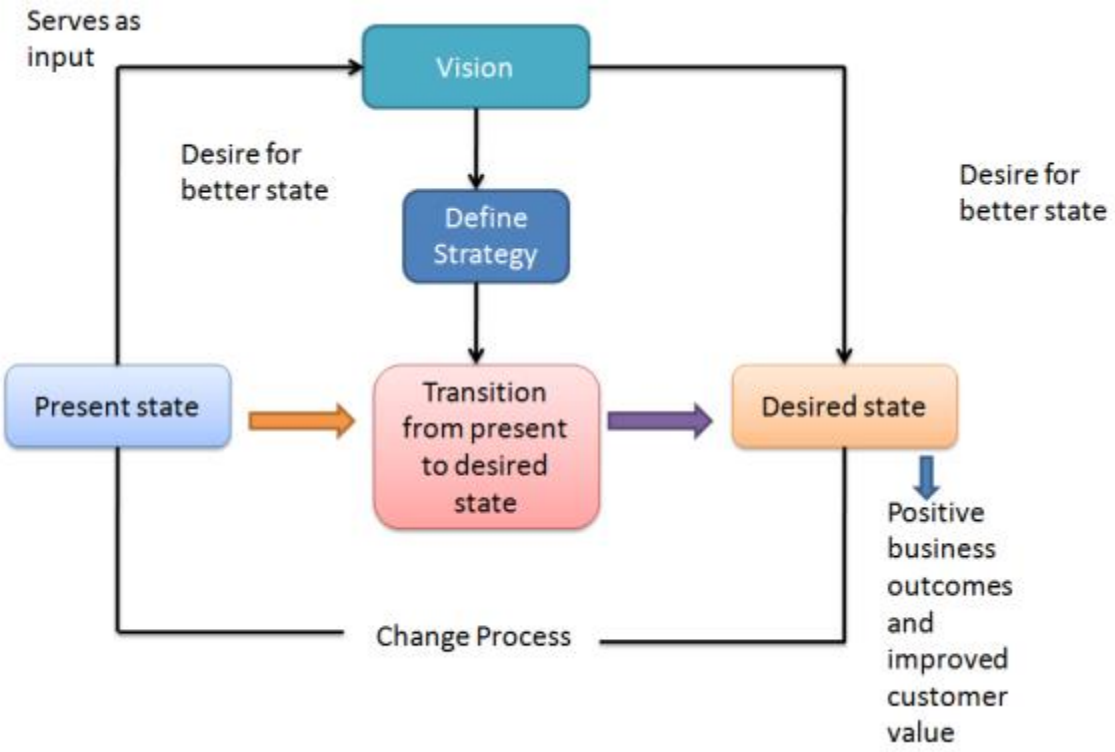


What Is Configuration Management?

What Is Change Management?

As the name suggests change management is all about dealing with change in the way we work. Change management is required at various levels right from company's level to individual level. Change management deals with:

1. Adapting to change: Many times it becomes extremely important to implement structured processes to bring about a positive change in the business. In present times of recession or slowdown of economy companies have to implement stable mechanisms that would help staff members deal with pressure in difficult conditions.
2. Controlling change: deal with the new changes in the business environment. No matter what new changes you make to the environment it is important to monitor changes and track the details of the system.
3. Effecting Change: generating profit from the positive changes.



What Is Change Management?

THANK YOU!

Software Testing is a complex, and yet an interesting subject. One of the biggest challenges with learning Software testing was to find useful training material which can really teach you Software testing.

And this was the main reason and motivation why International Software Test Institute™ wrote this book for you. We hope that you enjoyed reading this book as much as we had enjoyed while we were writing it and you managed to build a Software Testing foundation for yourself!

Please don't hesitate to e-mail any feedback to info@test-institute.org

If you would like to complement your Software Testing knowhow with Accredited Software Testing Certification Programs, please check out International Software Test Institute ™ homepage at http://www.test-institute.org

Our one-of-a-kind industry leading registration, examination and certification process is very simple, quick and completely online. You can find all details under the following link: http://www.test-institute.org/Accredited_Software_Test_Manager_Certification_ASTMC_Program.php

Afterwards, please feel free to do your registration from the following link: http://www.test-institute.org/Register_Software_Testing_Certification_Program.php