CHAPTER I

1.1 What is software testing? Why is it important?

A software or computer software essentially is a type of programs which enable the users to perform some particular task or actually used to operate their computer. It essentially directs all of the peripheral devices on the entire computer system- what exactly to do and how exactly to perform a task. A software plays a key role of a mediator between the user and the computer hardware.

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. In simple terms, Software Testing means the Verification of Application Under Test (AUT). The testing process involves evaluating the features of the software product for requirements in terms of any missing requirements, bugs or errors, security, reliability and performance. It also involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Why is Software Testing Important?

Software Testing is important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

Software testing is very important as many benefits. Some of the benefits are listed below:

- **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
- **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

- **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Who is responsible for software testing?

Software Quality Analyst/ Testers are responsible for the testing quality of software. They are involved in performing automated and manual tests to ensure the software created by developers is fit for purpose. Some of the duties include analysis of software, and systems, mitigate risk and prevent software issues.

Testing is required for an effective performance of software application or product and software QA is responsible to ensure it. It's important to ensure that the application does not result into any failure because it can be very expensive in the future or in the later stages of the development. Proper testing ensures that bugs and issues are detected early in the life cycle of the product or application. If defects related to requirements or design are detected late in the life cycle, it can be very expensive to fix them since this might require redesign, re-implementation and retesting of the application. In case of a product organization or startup which has only one product, poor quality of software may result in lack of adoption of the product and this may result in losses which the business may not recover from. Hence, a QA is responsible to prevent all the defects and possible loss.

1.2 Difference between Project and Product.

Project and product sound similar and the two concepts are often confused with each other. However, in practice they are very different and require different skills, governance and mindsets. Some of the differences between project and product are listed below:

Project	**Product**
A project is designed with a clear definition of what needs to be delivered and by when.	A product is designed to continually create value for customers by solving their problems.
A project can also be defined as a task or set of tasks required to develop a unique service or product.	A product can also be defined as something that satisfies current demand in the market.
A project has a beginning and end date. The delivery team is disbanded when complete until the next project initiates, which could be years afterwards.	Products have no end date. It has more permanence, are living entities which we deliver quickly, iterate constantly, and are not something that we just walk away from.
Project is led by a project manager.	Product is led by a product manager.
Project is mainly output focused.	Product is mainly outcome focused.
A project normally has fixed requirement.	A product tend to have evolving requirement.
Values are delivered at the end date of the project.	Ongoing gradual delivery of value is the agenda of product.

Table 1: Difference between Project and Product.

1.3 Difference between Quality Assurance (QA) and Quality Control (QC).

Quality assurance (QA) and quality control (QC) are two terms that are often used interchangeably. Although similar, there are distinct differences between the two concepts.

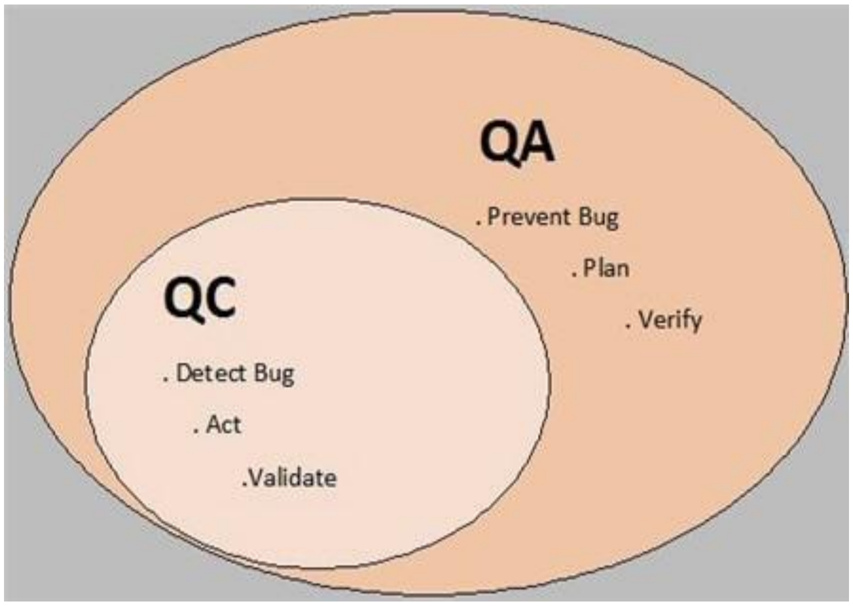


Figure 1- Quality Assurance (QA) Vs Quality Control (QC)

QA	**QC**
Quality Assurance (QA) is defined as an activity to ensure that an organization is providing the best possible product or service to customers.	Quality control is a Software Engineering process used to ensure quality in a product or a service. It does not deal with the processes used to create a product; rather it examines the quality of the "end products" and the final outcome.
It is a process which deliberates on providing assurance that quality request will be achieved.	QC is a process which deliberates on fulfilling the quality request.
It is a method to manage the quality-Verification.	It is a method to verify the quality-Validation.

QA ensures that everything is executed in the right way, and that is why it falls under verification activity.	QC ensures that whatever we have done is as per the requirement, and that is why it falls under validation activity.
A QA aim is to prevent the defect. Hence, it is a Preventive technique.	A QC aim is to identify and improve the defects. Hence, it is a Corrective technique.
QA involves in full software development life cycle.	QC involves in full software testing life cycle.
All team members are responsible for QA.	Testing team is responsible for QC.
In order to meet the customer requirements, QA defines standards and methodologies.	QC confirms that the standards are followed while working on the product.
It is the procedure to create the deliverables.	It is the procedure to verify that deliverables.
It is performed before Quality Control.	It is performed only after QA activity is done.

Table 2: Difference between Quality Assurance (QA) and Quality Control (QC).

"**If QA (Quality Assurance) is done then why do we need to perform QC (Quality Control)?"**

Well, this thought might come to testers mind, from time to time. If we have followed all the pre-defined processes, policies & standards correctly and completely then why do we need to perform a round of QC?

QC is required after QA is done. While doing 'QA', we define the processes, policies & strategies, establish standards, develop checklists etc. that needs to be used and followed throughout the life cycle of a project. And while doing QC we follow all those defined processes, standards and policies that we laid down in QA to make sure that the project is maintaining high quality and the final outcome of the project at least meets the customer’s expectations.

1.4. Roles and objectives of QA and QC

Software QA roles and objectives includes:

- Reviewing quality specifications and technical design documents to provide timely and meaningful feedback.
- Creating detailed, comprehensive and well-structured test plans and test cases.
- Estimating, prioritizing, planning and coordinating quality testing activities.
- Design, develop and execute automation scripts using open source tools.
- Identify, record, document thoroughly and track bugs.
- Perform thorough regression testing when bugs are resolved.
- Develop and apply testing processes for new and existing products to meet client needs.
- Co-ordinate with internal teams (e.g. developers and product managers) to identify system requirements.
- Monitor debugging process results.
- Investigate the causes of non-conforming software and train users to implement solutions.
- Track quality assurance metrics, like defect densities and open defect counts.
- Stay up-to-date with new testing tools and test strategies.

QA roles and objectives in agile methodology:

- Beyond the writing test cases, they should interact with other team members as an end user perspective from the starting of the project and in the absence of product owner they should keep the team moving forward as a proxy product owner.
- Participate in activities such as estimating stories by considering all the positive and negative scenarios and the previous testing experiences.

- Interactions with developer to understand how the functionality have been implemented so that the minor issues will be find out and it will be easy to do unit testing.
- Attend daily scrum and sprint-planning sessions.

Software QC roles and objectives includes:

- Gather and analyze quality control measurements (these are results of quality control activities used to review and evaluate the quality standards and product processes).
- Identify and audit non-conformance of project results.
- Suggest corrective and preventive actions and recommend quality improvements (through sending change requests being applied to the quality management plan).
- Validate procedures for fault elimination.
- Review the results of corrective actions and produce quality control reports.

1.5 Seven Testing Principles

Software testing is a process of executing a program with the aim of finding the error. To make our software perform well it should be error free. If testing is done successfully, it will remove all the errors from the software. Moreover, it is also necessary to understand the Seven Testing Principles, that every Software tester and QA professional should know, they are listed below:-

- Testing shows presence of defects
- Exhaustive testing is impossible
- Early testing
- Defect clustering
- Pesticide paradox
- Testing is context dependent
- Absence of errors fallacy

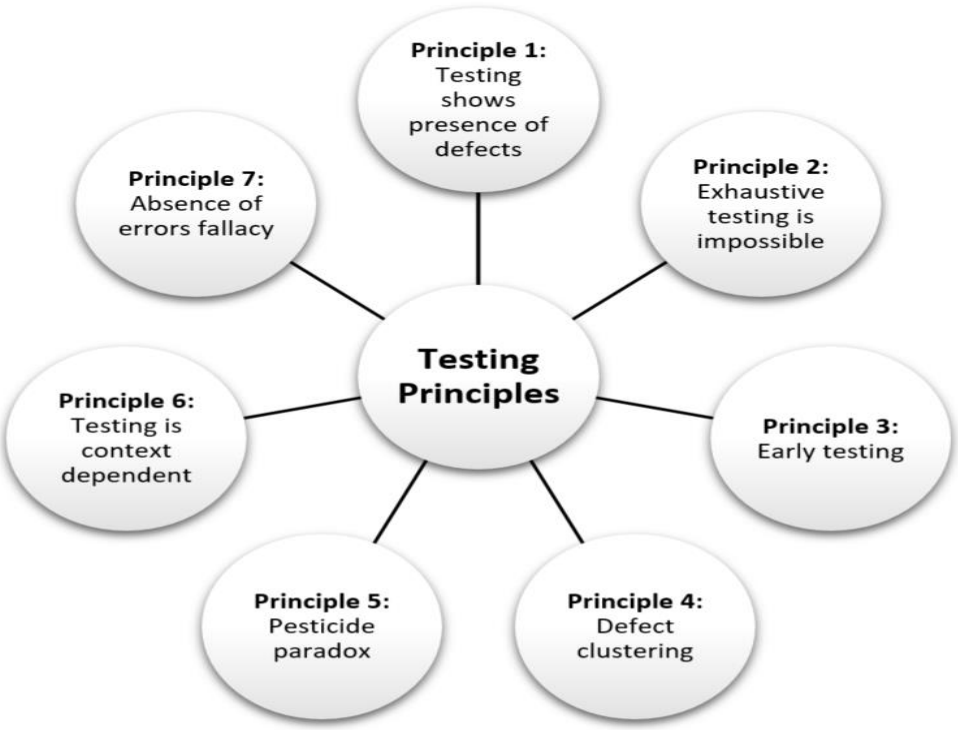

Principle 1: Testing shows presence of defects
Principle 2: Exhaustive testing is impossible
Principle 3: Early testing
Principle 4: Defect clustering
Principle 5: Pesticide paradox
Principle 6: Testing is context dependent
Principle 7: Absence of errors fallacy
Testing Principles

Figure 2- 7 testing principles

1) Exhaustive testing is impossible

Exhaustive testing (including or considering all element or aspects) is usually impossible. Instead, we need the perfect amount of testing based on the risky factors of the application. And the question is, how do we determine this risk? To answer this let us consider an example, Which operation is most likely to cause our Operating system to fail?

Opening 10 different application all at the same time would cause operating system to fail. So if we were testing this Operating system, we would realize that defects are likely to be found in multi-tasking activity and need to be tested thoroughly which brings us to our next principle Defect Clustering.

2) Defect Clustering

Defect Clustering states that a small number of modules contain most of the defects . This is the application of the “Pareto Principle” of software testing: Approximately 80% of the problems are found in 20% of the modules. By experience, you can identify such risky modules. But this approach has its own problems. Hence, If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.

3) Pesticide Paradox

Let us consider an example, When we use same pesticide to destroy insects during farming it will lead to insects to develop resistance to the pesticide which means ineffectiveness of pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects. To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects. Testers cannot simply depend on existing test techniques. He must look out continually to improve the existing methods to make testing more effective. But even after all this sweat & hard work in testing, you can never claim your product is bug-free.

4) Testing shows a presence of defects

Hence, this testing principle states that - Testing talks about the presence of defects and don’t talk about the absence of defects. i.e. Software Testing reduces the probability of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness. But what if, we work extra hard, taking all precautions & make our software product 99% bug-free. And the software does not meet the needs & requirements of the clients. This leads us to our next principle, which states that- Absence of Error

5) Absence of Error - fallacy

It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not only finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. Finding and fixing defects does not help if the system build is unusable and does not fulfill the user's needs & requirements. To solve this problem, the next principle of testing states that Early Testing

6) Early Testing

Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are detected in early stages. It is much cheaper to fix a Defect in the early stages of testing. It is recommended that we start finding the bug the moment the requirements are defined.

7) Testing is context dependent

Testing is context dependent which basically means the approaches depends on context of software developed. Different types of software needs different types of testing .For example, the way we test an e-commerce site will be different from the way you test a commercial off the shelf application (For eg :-Microsoft is a COTS software provider). All the developed software’s are not identical.We might use a different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine.

To sum up, we can conclude that it is important that we achieve optimum test results while conducting software testing without deviating from the goal. But how we determine that we are following the right strategy for testing or not. For that, we need to stick to these seven basic testing principles.

1.6. Verification Vs Validation

Verification

Verification in Software Testing is a process of checking documents, design, code, and program in order to identify if the software has been built according to the requirements or not. The main goal of verification process is to ensure quality of software application, design, architecture etc. The verification process involves activities like reviews, walk-throughs and inspection.

Validation

Validation in Software Testing is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not. The process helps to ensure that the software fulfills the desired use in an appropriate environment. The validation process involves activities like unit testing, integration testing, system testing and user acceptance testing.

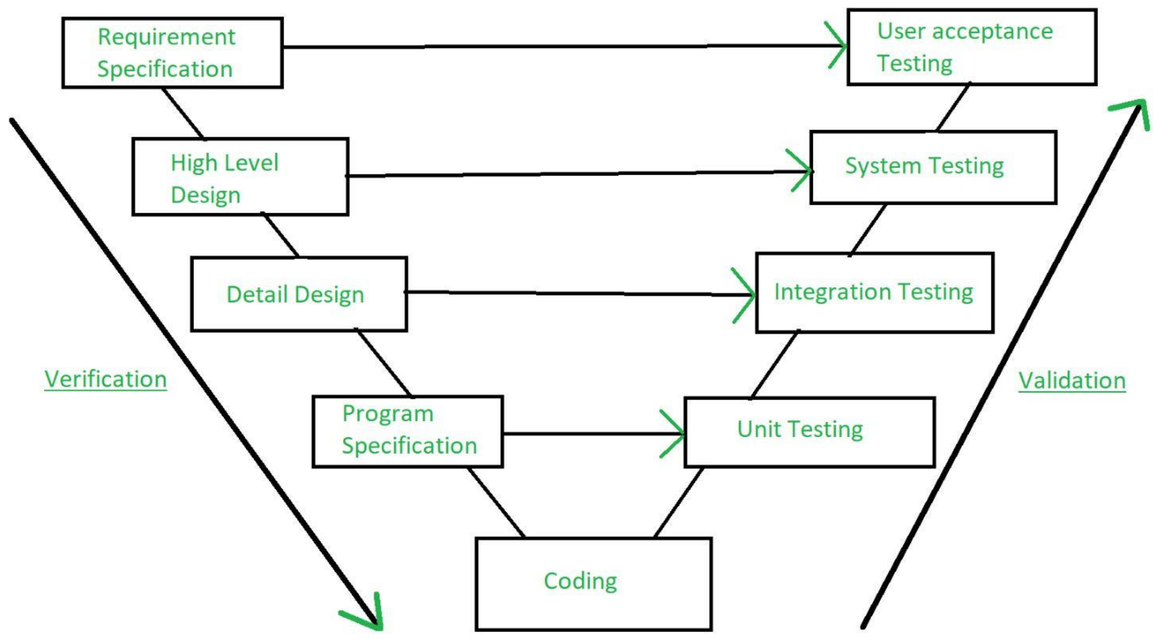


Figure 3- Verification and Validation

Verification VS Validation: Key Difference

Verification	Validation
The verifying process includes checking documents, design, code, and program	It is a dynamic mechanism of testing and validate the final end product like developed software or service or a system
It involves in verifying code without executing it.	It always involves executing the code.
Verification uses methods like reviews, walkthroughs, inspections, and desk- checking etc.	It uses methods like Black Box Testing, White Box Testing, and non-functional testing
It checks whether software confirms to requirements or not .	It checks whether the software meets the requirements and expectations of a customer
It finds bugs early in the development cycle	It can find bugs that the verification process cannot catch
Target is application and software architecture, specification, complete design, high level, and database design etc.	Target is an actual product
QA team does verification and make sure that the software is as per the requirement in the SRS document.	With the involvement of testing team validation is executed on software code.
It comes before validation	It comes after verification

Table 3: Difference Between validation and Verification

1.7. Bug Vs Defect Vs Error Vs Failure

Bug

If testers find any mismatch in the application/system in testing phase then they call it as Bug. However, there is a contradiction in the usage of Bug and Defect. People widely say the bug is an informal name for the defect.

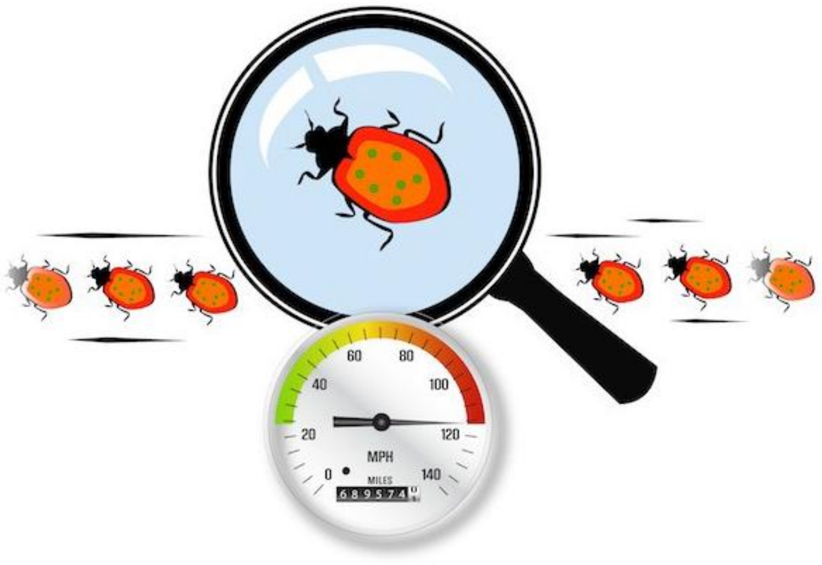


Figure 4- Bug

Defect

The variation between the actual results and expected results is known as defect. If a developer finds an issue and corrects it by himself in the development phase then it's called a defect.

Error

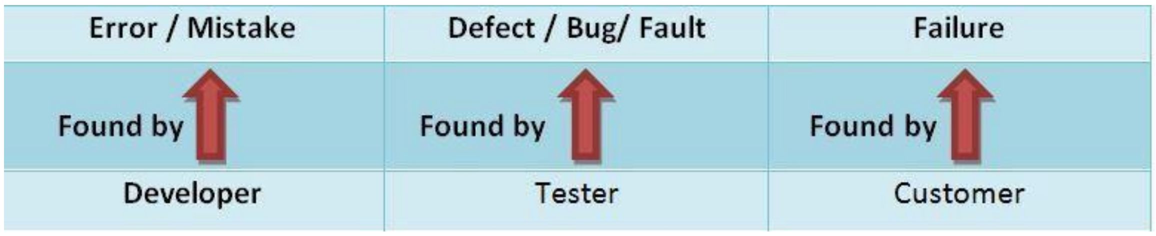
We can't compile or run a program due to coding mistake in a program. If a developer is unable to successfully compile or run a program then they call it as an **error.**

Failure

Once the product is deployed and customers find any issues then they call the product as a failure product. After release, if an end user finds an issue then that particular issue is called as failure**.**

Figure 5- Failure

Conclusion



Error / Mistake	Defect / Bug/ Fault	Failure
Found by ↑	Found by ↑	Found by ↑
Developer	Tester	Customer

Figure 6- Bug Vs defect Vs Error Vs Failure

A Bug is the result of a coding error and a defect is a deviation from the Requirements. A defect does not necessarily mean there is a bug in the code, it could be a function that was not implemented but defined in the requirements of the software.

1.8. Roles and Responsibilities of Actors

Business Analyst, Product owner, Project Managers, Scrum master, Developers, Designers, QA, Support team and Marketing team

- **Business Analyst**

The main task of business analyst is to identify customer business problems and find the most effective solution. Business Analyst works with the requirements at all stages of software development life cycle and constantly mediates between the customer and a team of programmers. They are involved in translating the solution features into software requirements. Then leads in analysis and designing phase, dictates in code development, then follows the testing phase during bug fixing as a change agent in the project team and ultimately fulfills the customer requirements.

Roles and Responsibilities:

- Identify customer needs, understand the problem he wants to solve.
- Develop idea independently or with a help of a team.
- Develop the idea into requirements specification to create future product. Different techniques of business analysis can be used: models of processes and structures, user interface prototypes, use cases. At the same time, it makes accurate estimates of efforts and work duration.
- Specify each requirement in specification form.
- Advise programmers and QA during product development and negotiate with customer any disputable issue.

- **Product Owner**

The product owner is the member of the Scrum team charged with maximizing the value of the team’s work. The product owner holds the product vision and works closely with stakeholders, such as end users, customers, and the business. They facilitate communication between the team and the stakeholders and ensure the team is building the right product. They describe what should be built and why, but not how.

Roles and Responsibilities:

- Defining the vision
- Managing the product backlog
- Prioritizing needs
- Overseeing development stages
- Anticipating client needs
- Acting as primary medium
- Evaluating product progress at each iteration

- **Project Managers**

  A software project manager defines the requirements of the project, builds the project team, lays out a blue print for the whole project including the project scope and parameters, clearly communicates the goals of the project to the team; the targets to be achieved, allocates budget to the various tasks to be completed, and ensures that the expectations of the Board of Directors and Stakeholders are met through timely completion of the project.

Roles and Responsibilities:

- Single Point of Contact (PoC) for Development Teams and the Client
- Uphold Quality, Consistency, and Product Delivery
- Ensure SDLC Standards are Upheld, Such as ISO or CMMI
- Document Everything
- Manage People and Teams

- **Scrum master**

  The scrum master is the team role responsible for ensuring the team lives agile values and principles and follows the processes and practices that the team agreed they would use.

  Roles and Responsibilities:

  - Clearing obstacles
  - Establishing an environment where the team can be effective
  - Addressing team dynamics
  - Ensuring a good relationship between the team and product owner as well as others outside the team
  - Protecting the team from outside interruptions and distractions.

- **Developers**
  - The Software Developers (front-end and back-end) are responsible for using the technical requirements from the Technical Lead to create cost and timeline estimates**.**
  - The Software Developers are also responsible for building the deliverables and communicating the status of the software project to the Technical Lead or Project Manager.
  - It is critical that the other team members effectively communicate the technical requirements to the Software Developers to reduce project risk and provide the software project with the greatest chance of success.

- **Designers**

UI designer focuses on the user's visual experience. It determines how a user interacts with an interface be it an app or a video game or a website. UI designer design all the screens through which a user will move, create the visual elements and their interactive properties.

- **QA**

It is always a good idea to include QA from the first phase of SDLC. In requirement understanding phase QA gets involved in document review of requirements (SRS). This will really help in understanding the requirement right from the early stages. In Design phase, the main task is to prepare the Test Plan, Test Plan along with test scenarios. During development phase QA will start writing the test cases. Test data, test scripts, test environment setup etc. Various testing types is performed during the test execution phase and defects are logged. We closely work with the development team in addressing the bug. During the maintenance phase test summary report is sent to all the stakeholders and we also participate in the lessons learnt sessions of the release.

- **Support team**

Support team maintains the computer networks of all types of organizations, providing technical support and ensuring the whole company runs smoothly. IT Support monitors and maintains the company computer systems, installs and configures hardware and software, and solves technical problems.

- **Marketing team**

The Marketing Department plays a vital role in promoting the business and mission of an organization. It is the Marketing Department's job to reach out to prospects, customers, investors and/or the community, while creating an overarching image that represents your company in a positive light.

1.9. What is SDLC?

Software is the collection of program and if you want to develop software to the customer than we need to go certain process called System Development Life Cycle, "SDLC" for short.

SDLC is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software which meets customer expectations.

ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

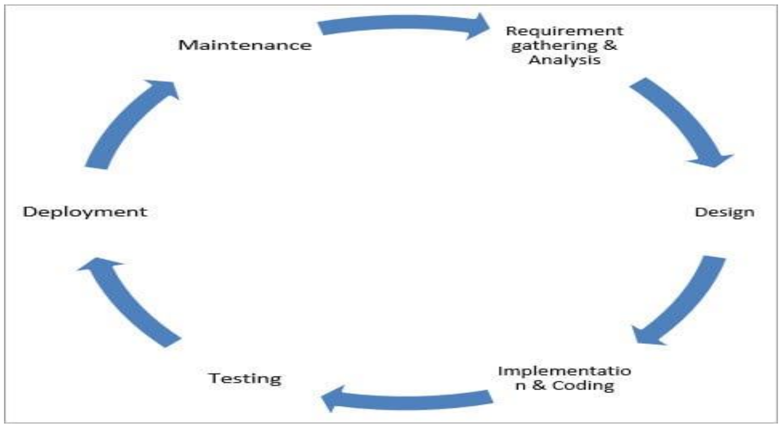


Figure 7 - SDLC Process

Given below are the various phases:

- Requirement gathering and analysis
- Design
- Implementation or coding
- Testing
- Deployment
- Maintenance

There are six phases in every Software development life cycle model:

1. **Requirement gathering and analysis**

Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.

For Example, A customer wants to have an application which involves money transactions. In this case, the requirement has to be clear like what kind of transactions will be done, how it will be done, in which currency it will be done, etc.

Finally, a Requirement Specification document is created which serves the

Purpose of guideline for the next phase of the model. The testing team follows the Software Testing Life Cycle and starts the Test Planning phase after the requirements analysis is completed.

2. **Design**

   In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. In this phase the testers comes up with the Test strategy, where they mention what to test, how to test.

3. **Implementation / Coding**

   On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

4. **Testing**

   Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

   During this phase all types of functional testing like unit testing, integration testing, system testing, acceptance testing are done as well as non-functional testing are also done.

5. **Deployment**

   Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

6. **Maintenance**

Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

1.9.1. Software Testing Life Cycle (STLC)

SDLC defines all the standard phases which are involved during the software development process, whereas the STLC process defines various activities to improve the quality of the product.

Software Testing Life Cycle refers to a testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met.

Below are the phases of STLC:

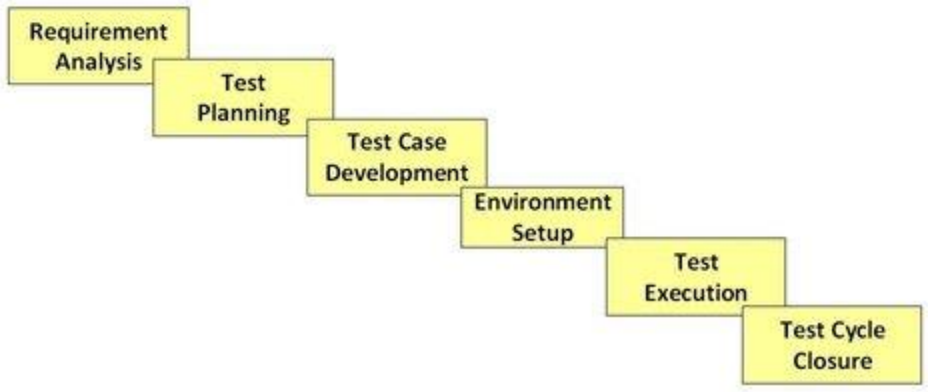


Figure 8 – STLC Phase

1. **Requirement Analysis**

During this phase, feature requirements collected in the SDLC process are evaluated to identify testable aspects and the QA team may interact with various stakeholders to understand requirements in detail.

Activities in Requirement Phase Testing Identify types of tests to be performed:

- Gather details about testing priorities and focus.
- Identify test environment details where testing is supposed to be carried out.
- Automation feasibility analysis (if required)

2. Test Planning

Typically, in this stage, a Senior QA manager will determine effort and cost estimates

For the project and would prepare and finalize the Test Plan.

Test Planning Activities:

- Preparation of test plan/strategy document for various types of testing.
- Test tool selection.
- Test effort estimation.
- Resource planning and determining roles and responsibilities.
- Training requirement

3. Test Case Development

This phase involves the creation, verification and rework of test cases and test scripts.

Test Case Development Activities done by QA team are:

- Create test cases, automation scripts (if applicable)
- Review test cases and scripts
- Create test data (If Test Environment is available)

4. Test Environment Setup

Test Environment Setup decides the software and hardware conditions under which a work product is tested. It is one of the critical aspects of the testing process and can be done in parallel with the Test Case Development Phase.

Test Environment Setup Activities:

- Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
- Setup test Environment and test data.

5. Test Execution

During this phase, the testers will carry out the testing based on the test plans and the test cases prepared. Expected test results are compared to actual and results are gathered to report back to development teams.

Test Execution Activities:

- Execute tests as per plan
- Document test results, and log defects for failed cases
- Track the defects to closure

6. Test Cycle Closure

This is the last phase of the STLC, during which a test result report is prepared.

Test Cycle Closure Activities:

- Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software, Critical Business Objectives and Quality.
- Document the learning out of the project.
- Prepare Test closure report
- Qualitative and quantitative reporting of quality of the work product to the customer.
- Test result analysis to find out the defect distribution by type and severity.

1.10. SDLC Models

There are various software development life cycle models defined and designed which are followed during the software development process.

Following are the most important and popular SDLC models followed in the industry:

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model
- Waterfall Model

1. Waterfall Model

Waterfall is the oldest and most straightforward of the structured SDLC methodologies; “finish one phase, then move on to the next.”

In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model:

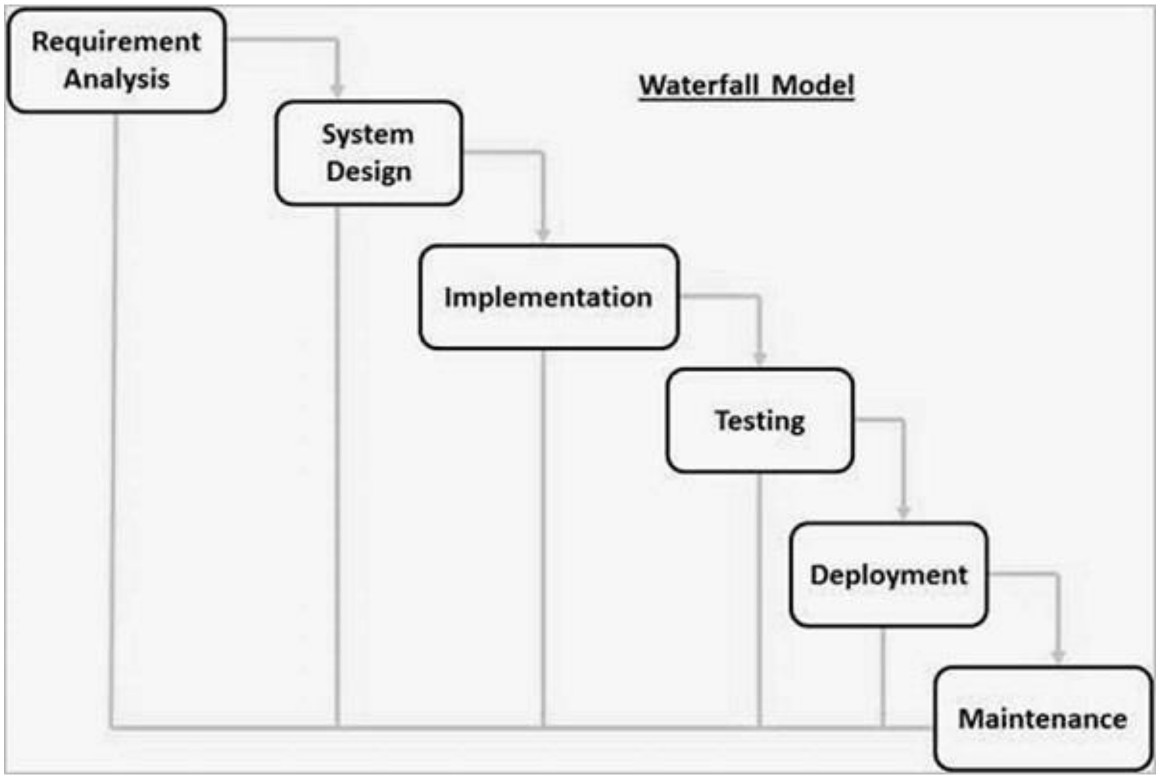


Figure 8 - Waterfall Model

The sequential phases in Waterfall model are:

i. **Requirement Gathering and analysis**

All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

ii. **System Design**

The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

iii. **Implementation**

With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

iv. **Integration and Testing**

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

v. **Deployment of system**

Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

vi. **Maintenance**

There are some issues which come up in the client environment. Also to enhance the product some better versions are released, maintenance is done to deliver these changes in the customer environment.

Here are the key pros and cons of Waterfall Model.

ADVANTAGES	**DISADVANTAGES**
Simple to use and understand	The software is ready only after the last stage is over
Management simplicity thanks to its rigidity: every phase has a defined result and process review	High risks and uncertainty
Development stages go one by one	Not the best choice for complex and object-oriented projects
Perfect for the small or mid-sized projects where requirements are clear	Inappropriate for the long-term projects

Easy to determine the key points in the development cycle	The progress of the stage is hard to measure while it is still in the development
Easy to classify and prioritize tasks	Integration is done at the very end, which does not give the option of identifying the problem in advance

Table.1 – Pros and Cons of Waterfall Model

2. Iterative Model

The Iterative SDLC model does not need the full list of requirements before the project starts. The development process may start with the requirements to the functional part, which can be expanded later. One advantage over other SDLC methodologies:

This model gives you a working version early in the process and makes it less expensive to implement changes. One disadvantage: Resources can quickly be eaten up by repeating the process again and again.

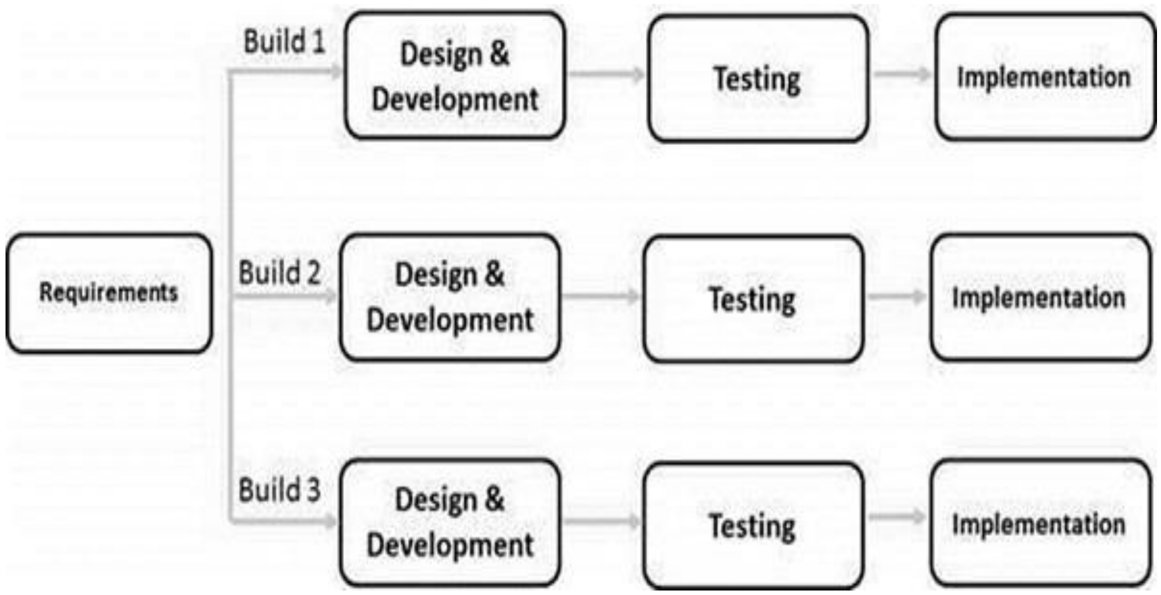


Figure 9- Iterative Model

3. Spiral Module

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model.

This model allows for the building of a highly customized product, and user feedback can be incorporated from early on in the project. But the risk you run is creating a never-ending spiral for a project that goes on and on.

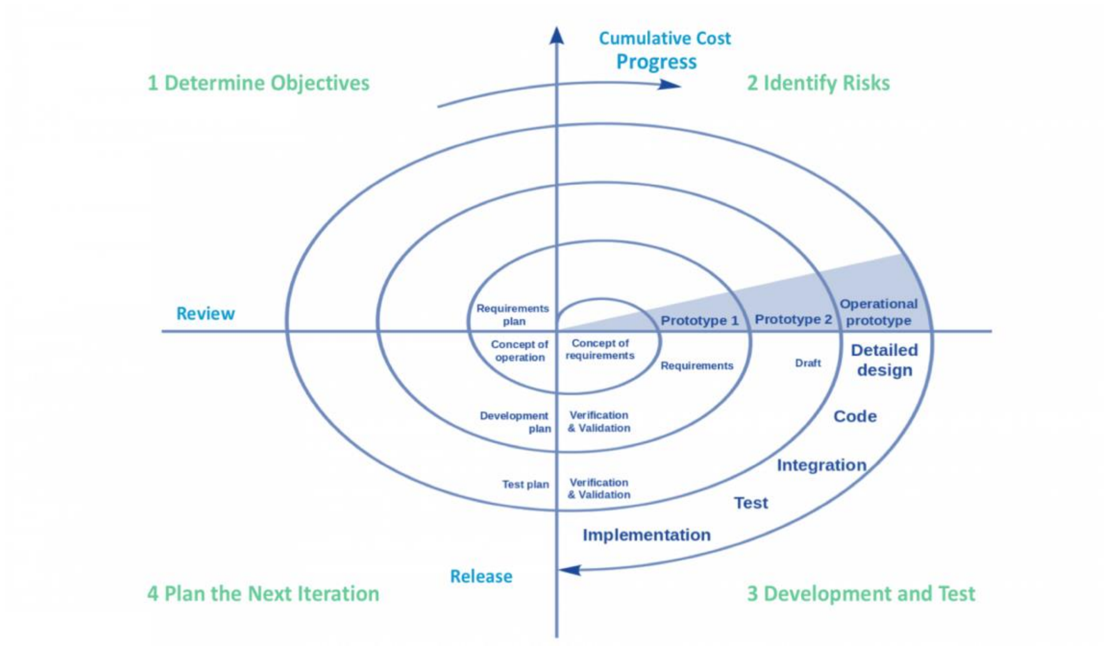


Figure 10- Spiral Module

4. V-Module

In this type of SDLC model testing and the development, the phase is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase

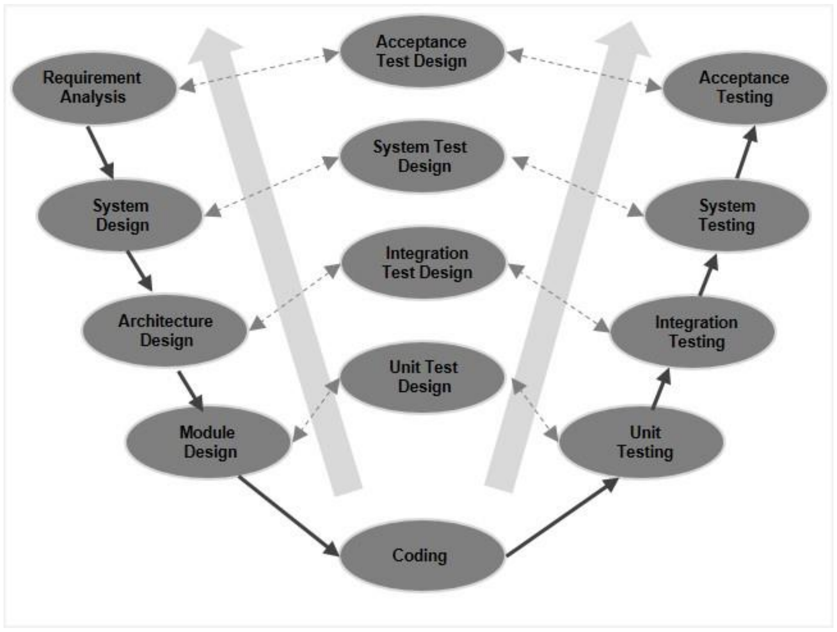


Figure 11- V Module

Verification Phases of V-Module

There are several Verification phases in the V-Model, each of these are explained in detail below:

i. **Business Requirement Analysis**

This is the first phase in the development cycle where the product requirements are understood from the customer's perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.

ii. System Design

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

iii. Architectural Design

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD).

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

iv. Module Design

In this phase, the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

v. Coding Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements.

The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

vi. Validation Phases

The different Validation Phases in a V-Model are explained in detail below:

a. Unit Testing

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

b. Integration Testing

Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

c. System Testing

System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

d. Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

ADVANTAGES	DISADVANTAGES
Every stage of V-shaped model has strict results so it’s easy to control	Lack of the flexibility
Testing and verification take place in the early stages	Bad choice for the small projects
Good for the small projects, where requirements are static and clear	Relatively big risks

Table.2 – Pros and cons of V-Module

5. Big bang model

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.

Big Bang is not recommended for large or complex projects, as it’s a high-risk model; if the requirements are misunderstood in the beginning, you could get to the end and realize the project may have to be started all over again.

1.11 Defect or Bug Life Cycle

A Defect life cycle, also known as a Bug life cycle, is a cycle of a defect from which it goes through covering the different states in its entire life. It starts when defect is found and ends when a defect is closed, after ensuring it’s not reproduced. Defect life cycle is related to the bug found during testing.

The defect life cycle can vary from organization to organization and also from project to project based on several factors like organization policy, software development model used (like Agile, Iterative), project timelines, team structure etc.

Some organizations / projects / managers may adopt a simpler life cycle while others may use a more extensive life cycle as described below.

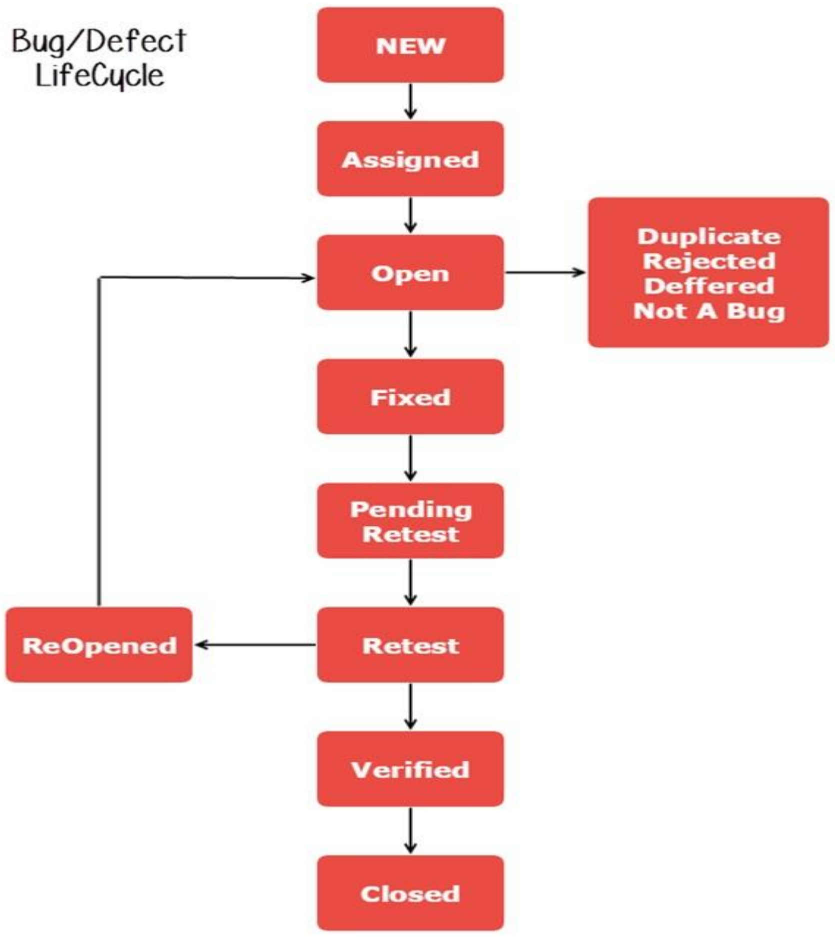


Figure 12 – Bug Life Cycle

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

1. New: When a defect is logged and posted for the first time. Its state is given as new.

2. Assigned: After the tester has posted the bug, the lead of the tester approves that the bug is genuine and he assigns the bug to corresponding developer and the developer team. Its state given as assigned.

3. Open: At this state the developer has started analyzing and working on the defect fix.

4. Fixed: When developer makes necessary code changes and verifies the changes then he/she can make bug status as ‘Fixed’ and the bug is passed to testing team.

5. Pending retest: After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.

6. Retest: At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.

7. Verified: The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to “verified”.

8. Reopen: If the bug still exists even after the bug is fixed by the developer, the tester changes the status to “reopened”. The bug goes through the life cycle once again.

9. Closed: Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to “closed”. This state means that the bug is fixed, tested and approved.

10. Duplicate: If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to “duplicate“.

11. Rejected: If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to “rejected”.

12. Deferred: The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are priority of the bug may be low, lack of time for the release or the bug may not have major effect on the software.

13. Not a bug: The state given as “Not a bug” if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and feel of the application like change of color of some text then it is not a bug but just some change in the look of the application.

1.12. Software Testing-Myths

There are various myths related to software testing. The 9 software testing myths are listed below:

- **Quality Control = Testing**

FACT: Testing is just one component of software quality control. Quality Control includes other activities such as Reviews.

- **The objective of Testing is to ensure a 100% defect- free product.**

FACT: The objective of testing is to uncover as many defects as possible while ensuring that the software meets the requirements. Identifying and getting rid of all defects is impossible

- **Testing is easy.**

FACT: Testing can be difficult and challenging (sometimes, even more so than coding).

- **Anyone can test**

FACT: Testing is a strict discipline and requires many kinds of skills.

- **There is no creativity in testing.**

FACT: Creativity can be applied when formulating test approaches, when designing tests, and even when executing tests.

- **Automated testing eliminates the need for manual testing.**

FACT: 100% test automation cannot be achieved. Manual Testing, to some level, is always necessary.

- **When a defect slips, it is the fault of the Testers.**

FACT: Quality is the responsibility of all members/ stakeholders, including developers, of a project.

- **Software Testing does not offer opportunities for career growth.**

FACT: Gone are the days when users had to accept whatever product was dished to them; no matter what the quality. With the abundance of competing software and increasingly demanding users, the need for software testers to ensure high quality will continue to grow. Software testing jobs are hot now.

References

- https://yourstory.com/mystory/what-software-types-examples
- https://www.guru99.com/software-testing-introduction-importance.html
- https://www.harbott.com/difference-between-product-and-project/
- https://www.guru99.com/quality-assurance-vs-quality-control.html
- https://www.guru99.com/software-testing-seven-principles.html
- https://www.edureka.co/blog/seven-principles-of-software-testing/
- https://www.geeksforgeeks.org/software-engineering-seven-principles-of-software-testing/
- https://www.javatpoint.com/software-testing-principles
- https://www.guru99.com/verification-v-s-validation-in-a-software-testing.html
- https://www.softwaretestingmaterial.com/difference-between-defect-bug-error-and-failure/
- https://medium.com/swlh/what-is-difference-between-defect-bug-error-
- https://resources.workable.com/qa-engineer-job-description
- https://mymanagementguide.com/quality-control-activities-ensuring-that-deliverables-comply-with-quality-requirements/
- https://blog.vsoftconsulting.com/blog/how-project-managers-add-value-to-the-sdlc
- https://project-management.com/project-manager-roles-responsibilities-software-projects/
- https://kishorsharma69.wordpress.com/2015/09/15/roles-of-qa-in-sdlc
- https://softwaretestingfundamentals.com/software-testing-myths/
- https://belitsoft.com/offshore-software-development-company/business-analyst-role
- https://www.lucidchart.com/blog/product-owner-roles-and-responsibilities
- http://gsa.github.io/tsd-agile-sdlc/scrum/roles-and-responsibilities/
- https://www.agilealliance.org/glossary/scrum-master/

- https://www.atlascode.com/blog/software-development-project-roles-and-responsibilities/#SOFTWARE_DEVELOPERS
- Guru99.com:https://www.guru99.com/what-is-sdlc-or-waterfall-model.html
- Tutorialspoint.com.: https://www.tutorialspoint.com/stlc/stlc_overview.htm>
- www.innovativearchitects.com: https://www.innovativearchitects.com/knowledgecenter/basic-it-systems/system-development-life-cycle.aspx
- linkedin.com:https://in.linkedin.com/jobs/view/mobile-application-tester-defect-bug-life-cycle-5-10-yrs-hyderabad-quality-assurance-at-lrr-technologies-2151759540