Testing Fundamentals

Khafidurrohman Agustianto, S.Pd., M.Eng.
guru99.com

Jurusan Teknologi Informasi
Politeknik Negeri Jember
2018

What is Software Testing? Introduction, Basics & Importance

What is Software Testing?

Software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves execution of a software component or system component to evaluate one or more properties of interest.

Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Some prefer saying Software testing as a white box and Black Box Testing.

This tutorial introduces testing software to the audience and justifies its importance

Please be patient. The Video will load in some time. If you still face issue viewing video click here

Why is Software Testing Important?

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, history is full of such examples.

- In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
- Nissan cars have to recall over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.
- Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point store served coffee for free as they unable to process the transaction.
- Some of the Amazon’s third party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.
- Vulnerability in Window 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system.
- In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.
- China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocent live
- In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.

- In April of 1999, a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history
- In may of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.

Types of Software Testing

Typically Testing is classified into three categories.

- Functional Testing
- Non-Functional Testing or Performance Testing
- Maintenance (Regression and Maintenance)

Testing Category	**Types of Testing**
Functional Testing	☐ Unit Testing ☐ Integration Testing ☐ Smoke ☐ UAT (User Acceptance Testing) ☐ Localization ☐ Globalization ☐ Interoperability ☐ So on
Non-Functional Testing	☐ Performance ☐ Endurance ☐ Load ☐ Volume ☐ Scalability ☐ Usability ☐ So on
Maintenance	☐ Regression ☐ Maintenance

This is not the complete list as there are more than 150 types of testing types and still adding. Also, note that not all testing types are applicable to all projects but depend on nature & scope of the project.

Software Testing as a Career Path (Skills, Salary, Growth)

This guide will take you through the In's and outs of software testing. If you plan to make a career in software testing, this is a MUST READ!

What is Software Testing?

Software Testing is a process of verifying a computer system/program to decide whether it meets the specified requirements and produces the desired results. As a result, you identify bugs in software product/project.

Software Testing is indispensable to provide a quality product without any bug or issue.

You will learn:

Skills required to become a Software Tester

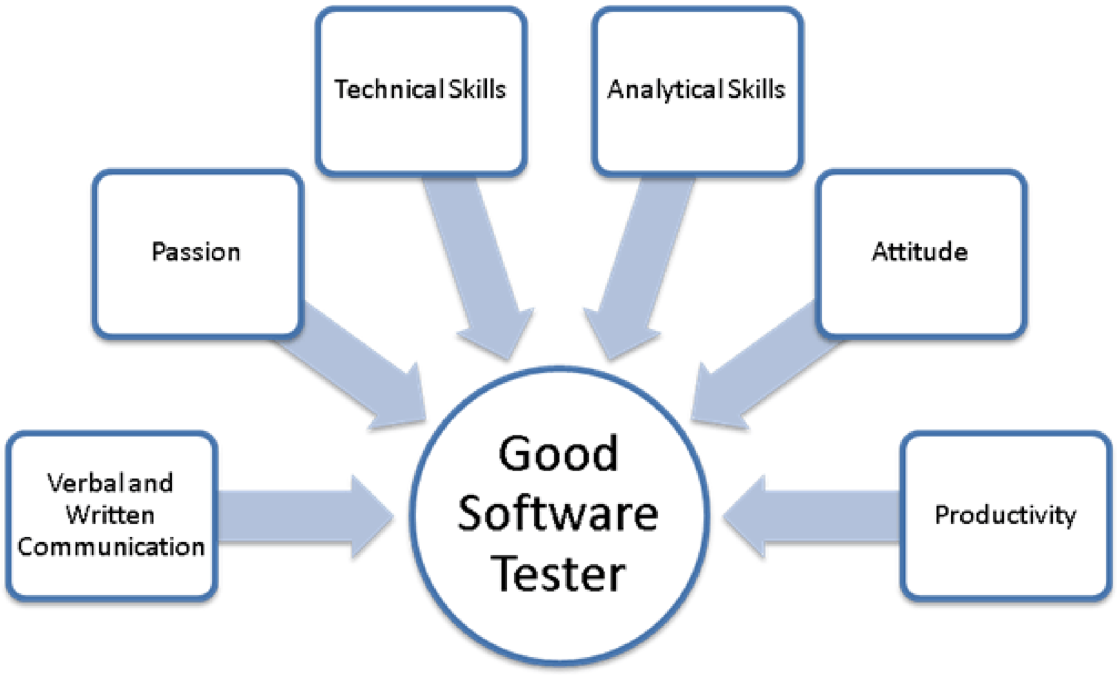
We will discuss the Technical and Non-Technical required to become a Software Tester

Non-Technical Skills

Following skills are essential to becoming a good software tester. Compare your skill set against the following checklist to determine whether Software Testing is a reality for you-

- **Analytical skills**: A good software tester should have sharp analytical skills. Analytical skills will help break up a complex software system into smaller units to gain a better understanding and create test cases. Not sure that you have good analytical skills - Refer this link - if, if you can solve at least ONE problem you have excellent analytical skills.

- **Communication skill**: A good software tester must have good verbal and written communication skill. Testing artifacts (like test cases/plans, test strategies, bug reports, etc.) created by the software tester should be easy to read and comprehend. Dealing with developers (in the event of bugs or any other issue) will require a shade of discreetness and diplomacy.
- **Time Management & Organization Skills:** Testing at times could be a demanding job especially during the release of code. A software tester must efficiently manage workload, have high productivity, exhibit optimal time management, and organization skills
- **GREAT Attitude:** To be a good software tester you must have a GREAT attitude. An attitude to 'test to break', detail orientation, willingness to learn and suggest process improvements. In the software industry, technologies evolve with an overwhelming speed, and a good software tester should upgrade his/her technical skills with the changing technologies. Your attitude must reflect a certain degree of independence where you take ownership of the task allocated and complete it without much direct supervision.
- **Passion:** To Excel in any profession or job, one must have a significant degree of the passion for it. A software tester must have a passion for his / her field. BUT how do you determine whether you have a passion for software testing if you have never tested before? Simple TRY it out and if software testing does not excite you switch to something else that holds your interest.



Technical Skills

This list is long, so please bear with us

- **Basic knowledge of Database/ SQL:** Software Systems have a large amount of data in the background. This data is stored in different types of databases like

Oracle, MySQL, etc. in the backend. So, there will be situations when this data needs to be validated. In that case, simple/complex SQL queries can be used to check whether proper data is stored in the backend databases.

- **Basic knowledge of Linux commands:** Most of the software applications like Web-Services, Databases, Application Servers are deployed on Linux machines.So it is crucial for testers to have knowledge about Linux commands.
- **Knowledge and hands-on experience of a Test Management Tool:**Test Management is an important aspect of Software testing. Without proper test management techniques, software testing process will fail. Test management is nothing but managing your testing related artifacts.

  For example - A tool like Testlink can be used for tracking all the test cases written by your team.

  There are other tools available that can be utilized for Test Management. So, it is important to have knowledge and working experience of such tools because they are used in most of the companies.

- **Knowledge and hands-on experience of any Defect Tracking tool-** Defect Tracking and Defect life cycle are key aspects of software testing. It is extremely critical to managing defects properly and track them in a systematic manner. Defect tracking becomes necessary because the entire team should know about the defect including managers, developers, and testers. Several tools are used to lock defects including QC, Bugzilla, Jira, etc.
- **Knowledge and hands-on experience of Automation tool:** If if you see yourself as an "Automation tester" after a couple of years working on manual testing, then you must master a tool and get in-depth, hands-on knowledge of automation tools.

  Note - Only knowledge of any Automation tool is not sufficient to crack the interview, you must have good hands-on experience, so practice the tool of your choice to achieve mastery. Knowledge of any scripting language like VBScript, JavaScript, C# is always helpful as a tester if you are looking for a job into automation. Few companies also use Shell/Perl scripting, and there is a lot of demand for testers having knowledge of the same. Again, it will depend on the company and which tools are used by that company.

There is also a lot of scope for performance testing tools because applications need to be tested for their performance which is a part of non-functional testing.

That's it to technical knowledge. Please note you do not need ALL the technical skills listed above. The technical skill sets required vary with the Job Role and company processes.

Academic Background

Academic background of a software tester should be in Computer Science.

A BTech/ B.E., MCA, BCA, BSc- Computers, will land you a job quickly.

If you do not hold any of these degrees, then you must complete a software testing certification like ISTQB and CSTE which help you learn Software Development/ Test Life Cycle and other testing methodologies.

Remuneration

Compensation of a software tester varies from company to company. Average salary range of a software tester in the US is $45,993 - $74,935. Average salary range of a software tester in India is Rs 247,315 - Rs 449,111.

Also, a software tester is also given health insurance, bonuses, gratuity and other perks.

What Does a Software Tester do?

On any typical work day, you will be busy understanding requirement documents, creating test cases, executing test cases, reporting and re-testing bugs, attending review meetings and other team building activities.

Software Tester Career Path

Your career progression as a software tester (QA Analyst) in typical CMMI level 5 company will look like following but will vary from company to company

1. QA Analyst (Fresher)
2. Sr. QA Analyst (2-3 years' experience)
3. QA Team Coordinator (5-6 years' experience)
4. Test Manager (8-11 years' experience)
5. Senior Test Manager (14+ experience)

Alternate Career Tracks as a Software Tester

Once you have got your hand dirty in manual testing, you can pursue following specializations

- **Automation Testing**: As an Automation Test Engineer, you will be responsible for automating manual test case execution which otherwise could be time-consuming. Tools used IBM Rational Robot, Silk performer, and QTP
- **Performance Testing:** As a performance test engineer, you will be responsible for checking application responsiveness (time is taken to load, maximum load application can handle), etc. Tools used WEBLoad, Loadrunner.
- **Business Analyst**: A major advantages Testers have over Developers is that they have an end to end business knowledge. An obvious career progression for testers is to become a Business Analyst. As a Business Analyst, you will be responsible for analyzing and assessing your company's business model and workflows. As a BA, you will intergrate these models and workflows with technology.

Common Myths

Software Testing as a Career pays Less Developers are more respected as compared to Testers

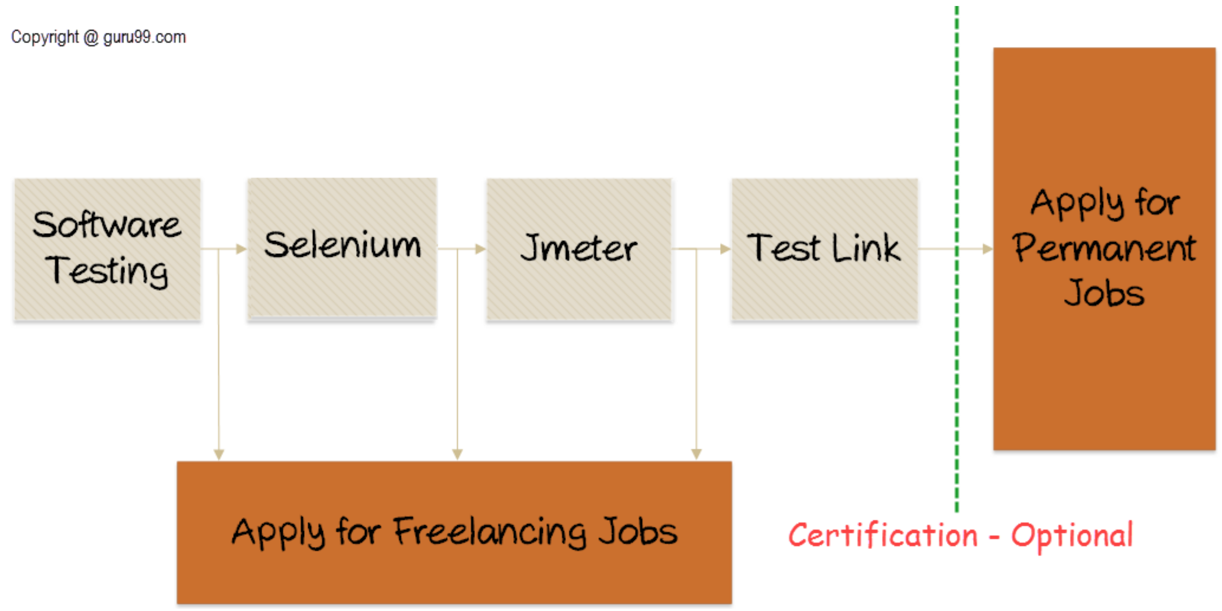
Contrary to popular belief, Software Testers (better known as QA professionals) are paid and treated at par with Software Developers in all "aspiring" companies. A career in Software Testing should never be considered as "second rated."

Software Testing is Boring

Software Testing could actually "test" your nerves since you need to make sense of Business Requirements and draft test cases based on your understanding. Software testing is not boring. What is boring is doing the same set of tasks repeatedly. The key is to try new things. For that matter, have you ever spoken to a software developer with more than 3 years' experience? He will tell you how boring his job has become off-lately.

How to Become Software Tester

For a complete newbie, here is our suggested approach to learning Software Testing



You start with learning Basic principles of Software Testing. Once done you **apply for freelancing jobs. This will help you gain practical knowledge and will fortify the testing concepts you have learned.**

Next, you proceed to Selenium - Automation tool, then Jemeter - Performance Testing tool and finally TestLink - Test Management Tool. All the while you are learning, we suggest you apply for freelancing jobs (apart from other benefits you will make some moolah too!).

Once you are through with all the tools, you may consider taking a certification. We recommend ISTQB. However, this is optional.

Certification Exams:

ISTQB Foundation level is the basic certification in Testing field.

It is not mandatory, but it will help increase your chances of getting the job. Most of the companies have this criterion.

A software tester with ISTQB cleared will be given more priority as compared to others.

After this, when you apply for permanent jobs in big corporations you will have many skills to offer as well some practical freelancing experience which may be of value and will increase your chances of being selected.

You can also pursue certification in a Testing tool of your choice.

Learning Guides: -

- Software Testing Tutorials - link
- Selenium - link As an alternative you can also learn QTP
- Jmeter - link As an alternative you can also learn Loadrunner
- Testlink - link As an alternative you can also learn Quality Center
- Freelancing Jobs – UpWork or Freelancer
- Permanent Jobs - Any major job portal like monster.com or naukri.com

7 Software Testing Principles: Learn with a Case Study

This tutorial introduces the seven basic principles of Software Testing every professional Software tester and QA professional should know.

Please be patient. The Video will load in some time. If you still face issue viewing video click here

Background

It is important that you achieve an optimum test results while conducting software testing without deviating from the goal. But how you determine that you are following right strategy for testing? For that, you need to stick to some basic testing principles. Here are the common seven testing principles that are widely practiced in the software industry.

To understand this, consider a scenario where you are moving a file from folder A to Folder B.

Think of all the possible ways you can test this.

Apart from the usual scenarios, you can also test the following conditions

Trying to move the file when it is Open

You do not have the security rights to paste the file in Folder B

Folder B is on a shared drive and storage capacity is full.

Folder B already has a file with the same name, infact the list is endless

Or suppose you have 15 input fields to test ,each having 5 possible values , the number of combinations to be tested would be 5^15

If you were to test the entire possible combinations project EXECUTION TIME & COSTS would rise exponentially. We need certain principles and strategies to optimize the testing effort

Here are the 7 Principles:

1) Exhaustive testing is not possible

Yes! Exhaustive testing is not possible. Instead, we need the optimal amount of testing based on the risk assessment of the application.

And the million dollar question is, how do you determine this risk ?

To answer this let's do an exercise

In your opinion, Which operation is most likely to cause your Operating system to fail?

I am sure most of you would have guessed, Opening 10 different application all at the same time.

So if you were testing this Operating system, you 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 which states that a small number of modules contain most of the defects detected. This is the application of the Pareto Principle to 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

If the same tests are repeated over and over again , eventually the same test cases will no longer find new bugs.

3) Pesticide Paradox

Repetitive use of the same pesticide mix to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective 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. To drive home this point , let's see this video of public launch of Windows 98

You think a company like MICROSOFT would not have tested their OS thoroughly & would risk their reputation just to see their OS crashing during its public launch!

4) Testing shows presence of defects

Hence, 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 , you work extra hard , taking all precautions & make your 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

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 mere finding defects, but also to check that software addresses the business needs. 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

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 captured in early stages. It is much cheaper to fix a Defect in early stages of testing. But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial.

7) Testing is context dependent

Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software’s are not identical. You 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.

Summary of the Seven Testing Principles

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

Myth: "Principles are just for reference. I will not use them in practice"

This is so very untrue. Test Principles will help you create an effective Test Strategy and draft error catching test cases.

But learning testing principles is just like learning to drive for the first time.

Initially while you learn to drive, you pay attention to each and everything like gear shifts, speed, clutch handling, etc. But with experience, you just focus on driving the rest comes naturally. Such that you even hold conversations with other passengers in the car.

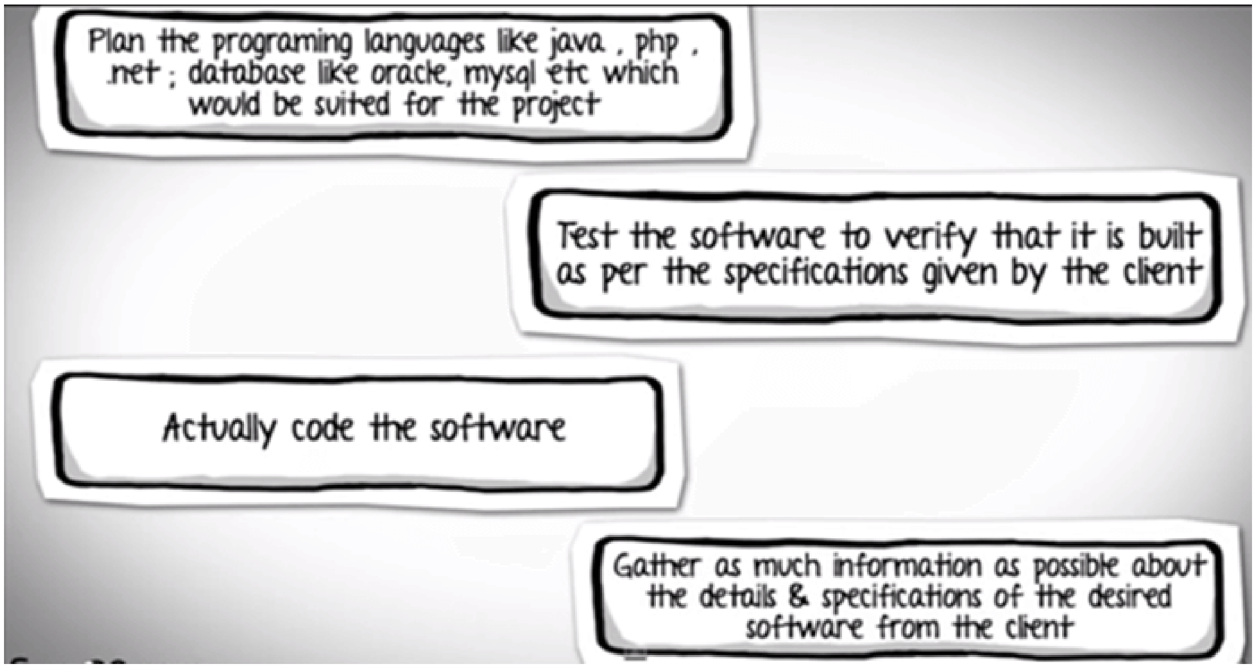
Same is true for testing principles. Experienced testers have internalized these principles to a level that they apply them even without thinking. Hence the myth that the principles are not use in practise is simply not true.

What is V Model? Learn with a Case Study using SDLC & STLC

This tutorial explains in detail the Software/System Development Life Cycle (SDLC) like the **Waterfall cycle & Iterative cycle like RAID & Agile**. And further, it proceeds to explain the V-Model of testing and STLC (Software Test Life Cycle).

Please be patient. The Video will load in some time. If you still face issue viewing video click here

Suppose, you are assigned a task, to develop a custom software for a client. Now, irrespective of your technical background, try and make an educated guess about the sequence of steps you will follow, to achieve the task.


Plan the programing languages like java , php , .net ; database like oracle, mysql etc which would be suited for the project
Test the software to verify that it is built as per the specifications given by the client
Actually code the software
Gather as much information as possible about the details & specifications of the desired software from the client

The correct sequence would be.

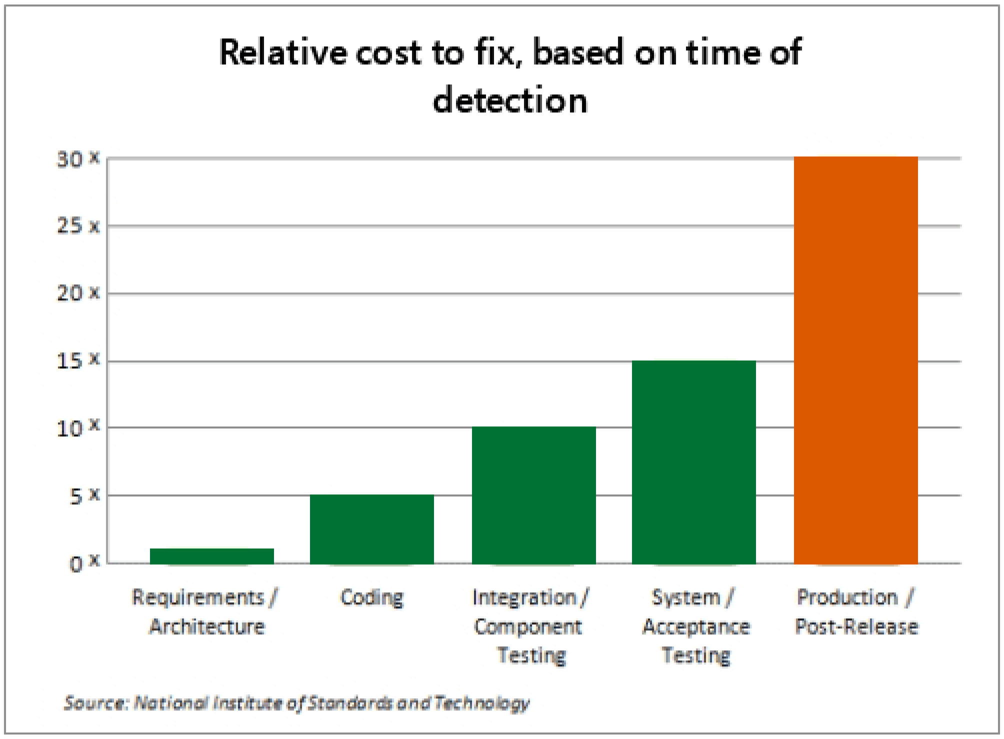
Different phases of Software Development Cycle	**Activities performed in each stage**
Requirement Gathering stage	• Gather as much information as possible about the details & specifications of the desired software from the client. This is nothing but the Requirements gathering stage.

Design Stage	• Plan the programming language like Java, PHP, .net; database like Oracle, MySQL, etc. Which would be suited for the project, also some high-level functions & architecture.
Build Stage	• After design stage, it is build stage, that is nothing but actually code the software
Test Stage	• Next, you test the software to verify that it is build as per the specifications given by the client.
Deployment stage	• Deploy the application in the respective environment
Maintenance stage	• Once your system is ready to use, you may require to change the code later on as per customer request

All these levels constitute the **waterfall method** of software development lifecycle. As you may observe, that **testing in the model starts only after implementation is done**.

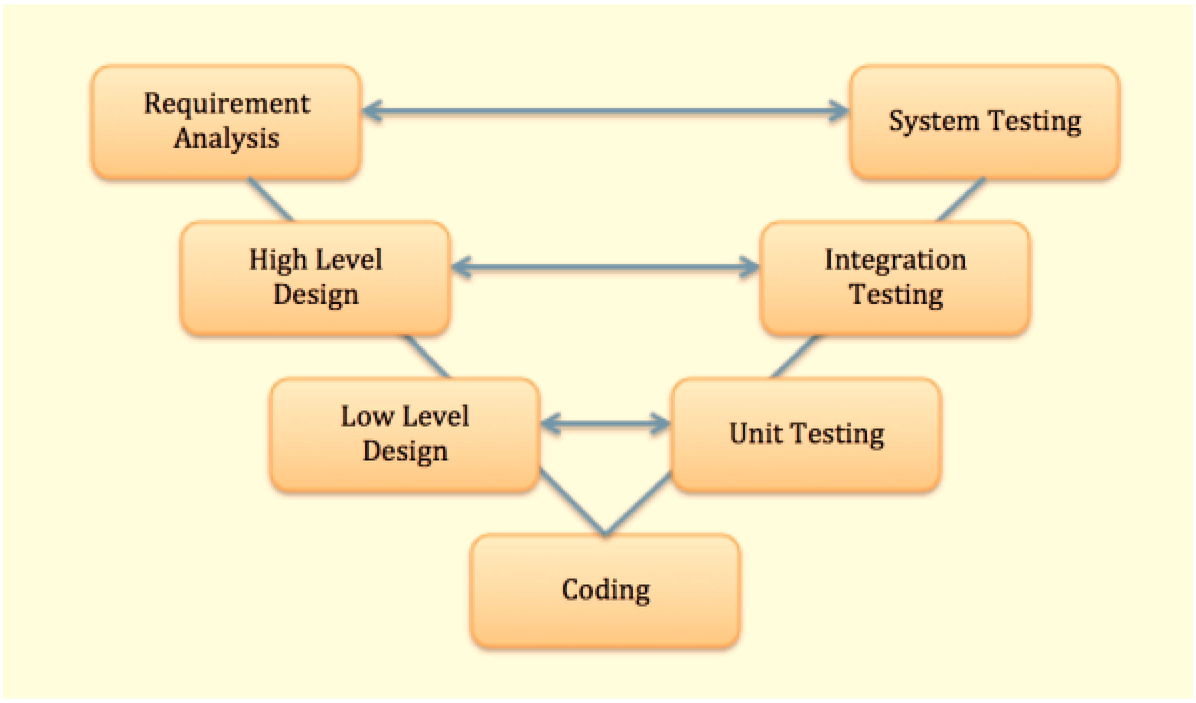
But if you are working in the large project, where the systems are complex, it's easy to miss out the key details in the requirements phase itself. In such cases, an entirely wrong product will be delivered to the client and you might have to start afresh with the project OR if you manage to note the requirements correctly but make serious mistakes in design and architecture of your software you will have to redesign the entire software to correct the error.

Assessments of thousands of projects have shown that **defects introduced during requirements & design make up close to half of the total number of defects.**



Also, the **costs of fixing a defect increases across the development life cycle**. **The earlier in life cycle a defect is detected, the cheaper it is to fix it.** As the say, "A stitch in time saves a nine."

To address this concern, **the V model of testing** was developed where **for every phase, in the Development life cycle there is a corresponding Testing phase**



- The left side of the model is Software Development Life Cycle - **SDLC**
- The right side of the model is Software Test Life Cycle - **STLC**
- The entire figure looks like a V, hence the name **V - model**

Apart from V model, there are iterative development models, where development is carried in phases, with each phase adding a functionality to the software. Each phase comprises of its independent set of development and testing activities.

Good examples of Development lifecycles following iterative method are Rapid Application Development, Agile Development

Here are the Key Terms again:

- **SDLC:**

  SDLC is Software Development Life Cycle. It is the sequence of activities carried out by Developers to design and develop high-quality software.

  Though SDLC uses the term ‘Development’, it does not involve just coding tasks done by developers but also incorporates the tasks contributed by testers and stakeholders.

  In SDLC, test cases are created.

- **STLC:**

  STLC is Software Testing Life Cycle. It consists of series of activities carried out by Testers methodologically to test your software product.

  Though STLC uses the term “testing” it does not involve just testers, at some instances, they have to involve developers as well.

  In STLC Test cases are executed.

- **Waterfall Model:**

  Waterfall model is a sequential model divided into different phases of software development activity. Each stage is designed for performing specific activity during SDLC phase. Testing phase in waterfall model starts only after implementation of the system is done.

  Testing is done within the SDLC.

- **V- Model:**

  V- model is an extension of the waterfall model. Unlike waterfall model, In V-model, there is a corresponding testing phase for each software development phase. Testing in V-model is done in parallel to SDLC stage.

  Testing is done as a sub project of SDLC.

Conclusion

There are numerous development life cycle models. **Development model selected for a project depends on the aims and goals of that project.**

- Testing is not a stand-alone activity, and it has to adapt the development model chosen for the project.
- In any model, testing should performed at all levels i.e. right from requirements until maintenance.

STLC - Software Testing Life Cycle

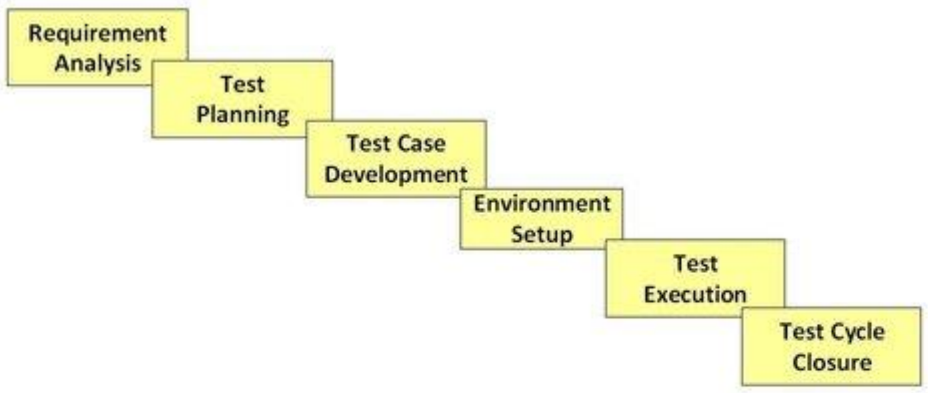
Contrary to popular belief, Software Testing is not a just a single activity.

What is Software Testing Life Cycle (STLC)?

Software Testing Life Cycle (STLC) is defined as a sequence of activities conducted to perform Software Testing.

It consists of series of activities carried out methodologically to help certify your software product.

Diagram - Different stages in Software Test Life Cycle



Each of these stages have a definite Entry and Exit criteria; , Activities & Deliverables associated with it.

What is Entry and Exit Criteria?

Entry Criteria: Entry Criteria gives the prerequisite items that must be completed before testing can begin.

Exit Criteria: Exit Criteria defines the items that must be completed before testing can be concluded

You have Entry and Exit Criteria for all levels in the Software Testing Life Cycle (STLC)

In an Ideal world you will not enter the next stage until the exit criteria for the previous stage is met. But practically this is not always possible. So for this tutorial, we will focus on activities and deliverables for the different stages in STLC life cycle. Lets look into them in detail.

Requirement Analysis

During this phase, test team studies the requirements from a testing point of view to identify the testable requirements.

The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, System Architects etc) to understand the requirements in detail.

Requirements could be either Functional (defining what the software must do) or Non Functional (defining system performance /security availability)

Automation feasibility for the given testing project is also done in this stage.

Activities

- Identify types of tests to be performed.
- Gather details about testing priorities and focus.
- Prepare Requirement Traceability Matrix (RTM).
- Identify test environment details where testing is supposed to be carried out.
- Automation feasibility analysis (if required).

Deliverables

- RTM
- Automation feasibility report. (if applicable)

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. In this phase, Test Strategy is also determined.

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

Deliverables

- Test plan /strategy document.
- Effort estimation document.

Test Case Development

This phase involves creation, verification and rework of test cases & test scripts. Test data , is identified/created and is reviewed and then reworked as well.

Activities

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

Deliverables

- Test cases/scripts
- Test data

Test Environment Setup

Test environment decides the software and hardware conditions under which a work product is tested. Test environment set-up is one of the critical aspects of testing process and ***can be done in parallel with Test Case Development Stage***. ***Test team may not be involved in this activity*** if the customer/development team provides the test environment in which case the test team is required to do a readiness check (smoke testing) of the given environment.

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
- Perform smoke test on the build

Deliverables

- Environment ready with test data set up
- Smoke Test Results.

Test Execution

During this phase the testers will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed.

Activities

- Execute tests as per plan
- Document test results, and log defects for failed cases
- Map defects to test cases in RTM
- Retest the Defect fixes
- Track the defects to closure

Deliverables

- Completed RTM with execution status
- Test cases updated with results
- Defect reports

Test Cycle Closure

Testing team will meet , discuss and analyze testing artifacts to identify strategies that have to be implemented in future, taking lessons from the current test cycle. The idea is to remove the process bottlenecks for future test cycles and share best practices for any similar projects in future.

Activities

- Evaluate cycle completion criteria based on Time,Test coverage,Cost,Software,Critical Business Objectives , Quality
- Prepare test metrics based on the above parameters.
- 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.

Deliverables

- Test Closure report
- Test metrics

Finally, ***summary*** of STLC Phases along with Entry and Exit Criteria

STLC Stage	Entry Criteria	Activity	Exit Criteria	Deliverables
Requirement Analysis	Requirements Document available (both functional and non functional)	Analyse business functionality to know the business modules and module specific functionalities.	Signed off RTM Test automation feasibility report signed off by the	RTM Automation feasibility report (if applicable)

STLC Stage	Entry Criteria	Activity	Exit Criteria	Deliverables
	Acceptance criteria defined. Application architectural document available.	Identify all transactions in the modules. Identify all the user profiles. Gather user interface/ authentication, geographic spread requirements. Identify types of tests to be performed. Gather details about testing priorities and focus. Prepare Requirement Traceability Matrix (RTM). Identify test environment details where testing is supposed to be carried out. Automation feasibility analysis (if required).	client	
Test Planning	Requirements Documents Requirement Traceability matrix. Test automation feasibility document.	Analyze various testing approaches available Finalize on the best suited approach Preparation of test plan/strategy document for various types of testing Test tool selection Test effort estimation Resource planning and determining roles and responsibilities.	Approved test plan/strategy document. Effort estimation document signed off.	Test plan/strategy document. Effort estimation document.

STLC Stage	Entry Criteria	Activity	Exit Criteria	Deliverables
Test case development	Requirements Documents RTM and test plan Automation analysis report	Create test cases, test design, automation scripts (where applicable) Review and baseline test cases and scripts Create test data	Reviewed and signed test Cases/scripts Reviewed and signed test data	Test cases/scripts Test data
Test Environment setup	System Design and architecture documents are available Environment set-up plan is available	Understand the required architecture, environment set-up Prepare hardware and software development requirement list Finalize connectivity requirements Prepare environment setup checklist Setup test Environment and test data Perform smoke test on the build Accept/reject the build depending on smoke test result	Environment setup is working as per the plan and checklist Test data setup is complete Smoke test is successful	Environment ready with test data set up Smoke Test Results.
Test Execution	Baselined RTM, Test Plan , Test case/scripts are available Test environment is ready Test data set up is done Unit/Integration test	Execute tests as per plan Document test results, and log defects for failed cases Update test plans/test cases, if necessary Map defects to test cases in RTM Retest the defect fixes	All tests planned are executed Defects logged and tracked to closure	Completed RTM with execution status Test cases updated with results Defect reports

STLC Stage	Entry Criteria	Activity	Exit Criteria	Deliverables
	report for the build to be tested is available	Regression Testing of application Track the defects to closure		
Test Cycle closure	Testing has been completed Test results are available Defect logs are available	Evaluate cycle completion criteria based on - Time, Test coverage , Cost , Software Quality , Critical Business Objectives Prepare test metrics based on the above parameters. 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		