Software Testing Cheatsheet

Seven Principles

This tutorial introduces the seven basic principles governing Software Testing.

* 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 will rise exponentially.**
* Hence, one of the testing principles states that **EXHAUSTIVE testing is not possible**.**Instead, we need an 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 operations are 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 and needs to be tested thoroughly which brings us to our next principle **Defect Clustering which states that a small number of modules contain most of the defects detected.**
* 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**
* This is the another principle of testing called **"Pesticide Paradox"**
* **To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.**
* But even after all this sweat & hard work in testing, you can never claim you 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 O.S thoroughly & would risk their reputation just to see their O.S crashing during its public launch!
* Hence, testing principle states that - **Testing shows presence 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 is a Fallacy i.e. Finding and fixing defects does not help if the system build is unusable and does not fulfill the users needs & requirements**
* To fix this problem, the next principle of testing states that
* **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 as well. More on this principle in a later training tutorial.
* And the last principle of testing states that the **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.**

**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 |

**Guide to SDLC , STLC & V-Model**

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).

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.

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](http://www.guru99.com/java-tutorial.html), [PHP](http://www.guru99.com/php-tutorials.html), .net; database like Oracle, MySQL, etc. Which would be suited for the project, also some high-level functions & architecture. |
| **Built Stage** | * After design stage, it is built stage, that is nothing but actually code the software |
| **Test Stage** | * Next, you test the software to verify that it is built 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 |

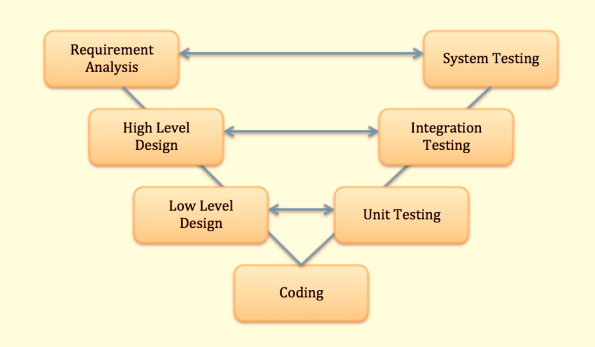
All these levels constitute the **waterfall method** of software development lifecycle. As you may observe, that **testing in the model starts only after implementation is done**.

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**

[](http://cdn.guru99.com/images/6-2015/052715_0904_GuidetoSDLC3.png)

* 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**

### 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.

# Software Testing Life Cycle STLC

Contrary to popular belief, Software Testing is not a just a single activity. It consists of series of activities carried out methodologically to help certify your software product. These activities (stages) constitute the Software Testing Life Cycle (STLC).

The different stages in Software Test Life Cycle -

[](http://cdn.guru99.com/images/stories/software-test-life-cycle.jpg)

Each of these stages have a definite [Entry and Exit criteria](http://www.guru99.com/faq.html#2)  , Activities & Deliverables associated with it.

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 of activities and deliverables for the different stages in STLC. 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)](http://www.guru99.com/traceability-matrix.html).
* 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

This phase is also called **Test Strategy** phase. 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.

### **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](http://www.guru99.com/test-plan.html) /strategy document.
* [Effort estimation](http://www.guru99.com/testing-estimation.html) document.

## Test Case Development

This phase involves creation, verification and rework of test cases & test scripts. [Test data](http://www.guru99.com/software-testing-test-data.html) , 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 test team 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 along with Entry and Exit Criteria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STLC Stage** | **Entry Criteria** | **Activity** | **Exit Criteria** | **Deliverables** |
| Requirement Analysis | Requirements Document available (both functional and non functional)  Acceptance criteria defined.  Application architectural document available. | Analyse business functionality to know the business modules and module specific functionalities.  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). | Signed off RTM  Test automation feasibility report signed off by the client | RTM  Automation feasibility report (if applicable) |
| 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. |
| Test case development | Requirements Documents  RTM and test plan  Automation analysis report | Create test cases, 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 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 report for the build to be tested is available | 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  Regression testing of application  Track the defects to closure | All tests planned are executed  Defects logged and tracked to closure | Completed RTM with execution status  Test cases updated with results  Defect reports |
| 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 | Test Closure report signed off by client | Test Closure report  Test metrics |

# Manual Testing

In Manual Testing , Testers manually execute test cases without using any automation tools. Manual testing is the most primitive of all testing types and helps find bugs in the software system.

 Any new application must be manually tested before its testing can be automated. Manual testing requires more effort, but is necessary to check  automation feasibility.

Manual Testing does not require knowledge of any testing tool. One of the  Software Testing Fundamental is "**100% Automation is not possible**". This makes Manual Testing imperative.

## Goal of Manual Testing

|  |  |
| --- | --- |
| [Manual Testing Tutorials for Beginners](http://cdn.guru99.com/images/goalofmanualtesting.png) | The goal of Manual Testing is to ensure that the application is error free and it is working in conformance to the specified functional requirements. Test Suites or cases ,are designed during the testing phase and should have 100% test coverage.  It also makes sure that reported defects are fixed by developers and re-testing has been performed by testers on the fixed defects. Basically, this testing checks the quality of the system and delivers bug-free product to the customer. |

## Manual Testing types**:**

Below given diagram depicts Manual Testing Types. In fact any type of software testing type can be executed both manually as well using an automation tool.



Myths of Manual Testing

Following are few common myths and facts related to testing:

Myth: Anyone can do manual testing

**Fact**: Testing requires many skill sets

Myth: Testing ensures 100% defect free product

**Fact**: Testing attempts to find as many defects as possible. Identifying all possible defects is impossible.

Myth: Automated testing is more powerful than manual testing

**Fact**: 100% test automation cannot be done. Manual Testing is also essential.

Myth: Testing is easy

**Fact**: Testing can be extremely challenging .Testing an application for possible use cases with minimum test cases requires high analytical skills.

## Comparison of Manual and Automated Testing

Automation Testing is use of tools to execute test cases whereas manual testing requires human intervention for test execution.

**Automation Testing saves time, cost and manpowe**r. Once recorded, it's easier to run an automated test suite when compared to manual testing which will require skilled labor.

**Any type of application can be tested manually but automated testing is recommended only for stable systems**and is mostly used for regression testing. Also, certain testing types like ad-hoc and monkey testing are more suited for manual execution.

**Manual testing can be become repetitive and boring**. On the contrary, the boring part of executing same test cases time and again, is handled by automation software in automation testing.

**Conclusion**

Manual testing is an activity where the tester needs to be very patient, creative &  open minded.They need to think and act with an End User perspective.