**Software testing life cycle** identifies what test activities to carry out and when (what is the best time) to accomplish those test activities. Even though testing differs between organizations, there is a testing life cycle.

**Software Testing Life Cycle** consists of six (generic) phases:

* Test Planning,
* Test Analysis,
* Test Design,
* Construction and verification,
* Testing Cycles,
* Final Testing and Implementation and
* Post Implementation.
* **Software testing** has its own life cycle that intersects with every stage of the SDLC. The basic requirements in**software testing life cycle** is to control/deal with software testing – Manual, Automated and Performance.
* **Test Planning**
* This is the phase where Project Manager has to decide what things need to be tested, do I have the appropriate budget etc. Naturally proper planning at this stage would greatly reduce the risk of low quality [software](http://www.testingbrain.com/ARTICLES/software%20testing%20life%20cycle.html). This planning will be an ongoing process with no end point.
* Activities at this stage would include preparation of high level test plan-(according to IEEE test plan template The[Software Test Plan](http://www.testingbrain.com/ARTICLES/software%20testing%20life%20cycle.html) (STP) is designed to prescribe the scope, approach, resources, and schedule of all testing activities. The plan must identify the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan.). Almost all of the activities done during this stage are included in this software test plan and revolve around a test plan.
* **Test Analysis**
* Once test plan is made and decided upon, next step is to delve little more into the project and decide what types of testing should be carried out at different stages of SDLC, do we need or plan to automate, if yes then when the appropriate time to automate is, what type of specific documentation I need for testing.
* Proper and regular meetings should be held between testing teams, project managers, development teams, Business Analysts to check the progress of things which will give a fair idea of the movement of the project and ensure the completeness of the test plan created in the planning phase, which will further help in enhancing the right testing strategy created earlier. We will start creating test case formats and test cases itself. In this stage we need to develop Functional validation matrix based on Business Requirements to ensure that all system requirements are covered by one or more test cases, identify which test cases to automate, begin review of documentation, i.e. Functional Design, Business Requirements, Product Specifications, Product Externals etc. We also have to define areas for Stress and [Performance testing](http://www.testingbrain.com/ARTICLES/software%20testing%20life%20cycle.html).
* **Test Design**
* Test plans and cases which were developed in the analysis phase are revised. Functional validation matrix is also revised and finalized. In this stage risk assessment criteria is developed. If you have thought of automation then you have to select which test cases to automate and begin writing scripts for them. Test data is prepared. Standards for unit testing and pass / fail criteria are defined here. Schedule for testing is revised (if necessary) & finalized and test environment is prepared.
* **Construction and verification**
* In this phase we have to complete all the test plans, test cases, complete the scripting of the automated test cases, Stress and Performance testing plans needs to be completed. We have to support the development team in their unit testing phase. And obviously bug reporting would be done as when the bugs are found. Integration tests are performed and errors (if any) are reported.
* **Testing Cycles**
* In this phase we have to complete testing cycles until test cases are executed without errors or a predefined condition is reached. Run test cases --> Report Bugs --> revise test cases (if needed) --> add new test cases (if needed) --> bug fixing --> retesting (test cycle 2, test cycle 3….).
* **Final Testing and Implementation**
* In this we have to execute remaining stress and performance test cases, documentation for testing is completed / updated, provide and complete different matrices for testing. Acceptance, load and recovery testing will also be conducted and the [application](http://www.testingbrain.com/ARTICLES/software%20testing%20life%20cycle.html) needs to be verified under production conditions.
* **Post Implementation**
* In this phase, the testing process is evaluated and lessons learnt from that testing process are documented. Line of attack to prevent similar problems in future projects is identified. Create plans to improve the processes. The recording of new errors and enhancements is an ongoing process. Cleaning up of test environment is done and test machines are restored to base lines in this stage.

|  |  |  |
| --- | --- | --- |
| |  | | --- | | Use risk analysis to determine where testing should be focused. Since it's rarely possible to test every possible aspect of an [application](javascript:void(0);), every possible combination of events, every dependency, or everything that could go wrong, risk analysis is appropriate to mostsoftware development projects. This requires judgement skills, common sense, and experience. Considerations can include: ? Which functionality is most important to the project's intended purpose. ? Which functionality is most visible to the user. ? Which functionality has the largest safety impact. ? Which functionality has the largest financial impact on users. ? Which aspects of the application are most important to the customer. ? Which aspects of the application can be tested early in the development cycle. ? Which parts of the code are most complex, and thus most subject to errors. ? Which parts of the application were developed in rush or panic mode. ? Which aspects of similar/related previous projects caused problems. ? Which aspects of similar/related previous projects had large maintenance expenses. ? Which parts of the requirements and design are unclear or poorly thought out. ? What do the developers think are the highest-risk aspects of the application. ? What kinds of problems would cause the worst [publicity](javascript:void(0);). ? What kinds of problems would cause the most [customer service](javascript:void(0);) complaints. ? What kinds of tests could easily cover multiple functionalities. ? Which tests will have the best high-risk-coverage to time-required ratio.  **Test Plan:** |  |  | | --- | |  |   [What if there is not enough time for thorough testing?](http://www.coolinterview.com/interview/21432/) |
| |  | | --- | | A [software](javascript:void(0);) [project](javascript:void(0);) test plan is a document that describes the objectives, scope, approach, and focus of asoftware testing effort. The process of preparing a test plan is a useful way to think through the efforts needed to validate the acceptability of a software product. The completed document will [help](javascript:void(0);) people outside the test group understand the 'why' and 'how' of product validation. It should be thorough enough to be useful but not so thorough that no one outside the test group will read it. The following are some of the items that might be included in a test plan, depending on the particular project: 1. Title 2. Identification of software including version/release numbers 3. Revision history of document including authors, dates, approvals 4. Table of Contents 5. Purpose of document, intended audience 6. Objective of testing effort 7. Software product overview 8. Relevant related document list, such as requirements, design documents, other test plans, etc. 9. Relevant standards or legal requirements 10. Traceability requirements 11. Relevant naming conventions and identifier conventions 12. Overall software project organization and personnel/contact-info/responsibilties 13. Test organization and personnel/contact-info/responsibilities 14. Assumptions and dependencies 15. Project risk analysis 16. Testing priorities and focus 17. Scope and limitations of testing 18. Test outline - a decomposition of the test approach by test type, feature, functionality, process, system, module, etc. as applicable 19. Outline of data input equivalence classes, boundary value analysis, error classes 20. Test environment - hardware, operating systems, other required software, data configurations, interfaces to other systems 21. Test environment validity analysis - differences between the test and production systems and their impact on test validity. 22. Test environment setup and configuration issues 23. Software migration processes 24. Software CM processes 25. Test data setup requirements 26. Database setup requirements 27. Outline of system-logging/error-logging/other capabilities, and tools such as screen capture software, that will be used to help describe and report bugs 28. Discussion of any specialized software or hardware tools that will be used by testers to help track the cause or source of bugs 29. Test [automation](javascript:void(0);) - justification and overview 30. Test tools to be used, including versions, patches, etc. 31. Test script/test code maintenance processes and version control 32. Problem tracking and [resolution](javascript:void(0);) - tools and processes 33. Project test metrics to be used 34. Reporting requirements and testing deliverables 35. Software entrance and exit criteria 36. Initial sanity testing period and criteria 37. Test suspension and restart criteria 38. Personnel allocation 39. Personnel pre-training needs 40. Test site/location 41. Outside test organizations to be utilized and their purpose, responsibilties, deliverables, contact persons, and coordination issues 42. Relevant proprietary, classified, [security](javascript:void(0);), and licensing issues. 43. Open issues | |