Software Testing Process – What Happens in Software Testing?

testbytes.net/blog/software-testing-process
January 31, 2019
Process

Thursday January 31, 2019

Understanding the Software Testing Process can be difficult even for the best of us. Discussed below is the basic template of the software testing process that is adapted by testers based on their particular requirements.

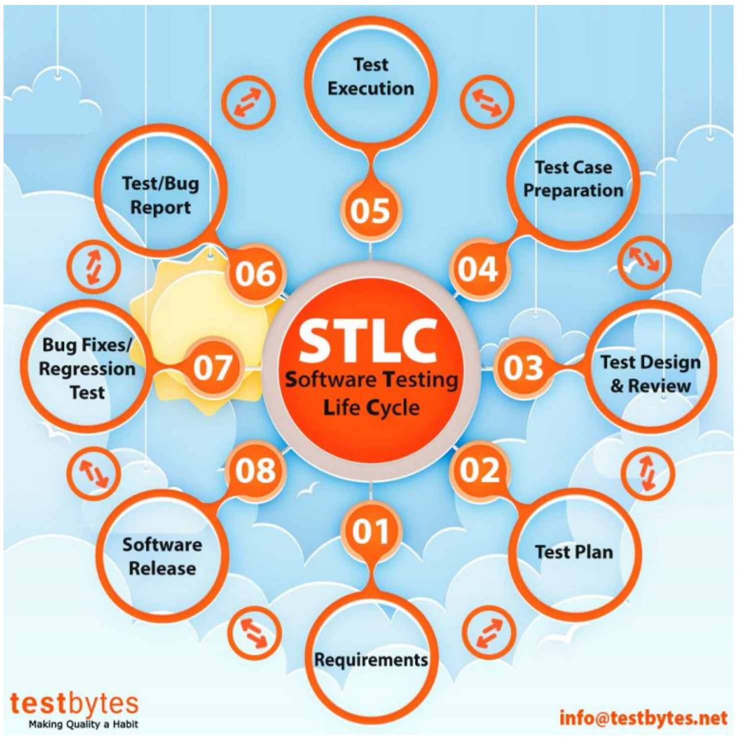
What is Software Testing?

Software Testing refers to the process of evaluating software and its components to identify any errors, bugs or errors that might potentially disrupt the functionality of the software.

It is essential to undertake a Software Testing Process as it bridges the gap between the existing and the required system software through the detection of the defects prior to the launch of the Software so that they can be corrected in time.

What are the Different Types of Software Testing Process?

Depending on the project requirements, budget associations and expertise of the team, Software Testing Process can be conducted in two ways



Manual Testing and Automation Testing.

1) Manual Testing

Manual Testing is the Software Testing Process that allows the tester to locate bugs or defects in the Software Program being tested.

The role of the tester is to use the software like the end user would, and then identify problems and mitigate them to ensure optimum functionality of the Software.

The tester is solely responsible for executing all test cases manually without turning to any automation tools.

The execution is undertaken by the tester preparing a test plan document detailing the systematic approach to Software Testing.

2) Automation Testing

Automation Testing is a technique that uses an application to undertake the implementation of the Software Testing Cycle in its entirety.

It uses scripted sequences executed by Testing Tools. It is a process that validates software functionality prior to the release of the Software into Production.

It is a way of simplifying manual efforts into a set of scripts that can be accessed and worked upon by the system. Automation Testing considerably reduces the time involved in the whole process of Software Testing while simultaneously enhancing the efficiency and effectiveness of the process.

Depending on the Software Testing Process that is followed, there are two major types of Software Testing.

These are discussed below.

1) Structured Software Testing

This is the kind of Software Testing wherein the tests and test cases are derived from a thorough knowledge of the structural code of the Software and its Internal Implementation.

Since it directly deals with the knowledge of the code, it is mostly undertaken by a trained team of developers.

2) Unstructured Software Testing

This is the kind of Software Testing that is performed without prior planning or documentation.

It is considered to be the least formal testing method and is only intended to run once unless of course an error is detected.

In that case, it is run repeatedly until the error is mitigated. Also known as Ad Hoc Testing, it is performed by Improvisation, as the sole aim is to detect a bug by taking up whatever means needed.

The following of the sequential steps that comprise the Structured Software Testing Life Cycle, ensure that standards are met with respect to the quality of the Software in question.

What are the Different Phases in the Structured Software Testing Life Cycle?

Requirement Analysis

The first step in the Software Testing Life Cycle is to identify which are the features of the Software that can be tested and how.

Any requirement of the Software that is revealed to be un-testable is identified at this stage, and subsequent mitigation strategies are planned. The Requirements that are arrived at here can either be Functional (related to the basic functions the software is supposed to do) in nature or Non-Functional (related to system performance or security availability).

Deliverables

- RTM – Requirement Traceability Matrix.
- Automation Feasibility Report

Test Planning

Now that the testing team has a list of requirements that are to be tested, the next step for them is to devise activities and resources, which are crucial to the practicality of the testing process. This is where the metrics are also identified, which will facilitate the supervision of the testing process. A senior Quality Assurance Manager will be involved at this stage to determine the cost estimates for the project. It is only after running the plan by the QA manager that the Test Plan will be finalized.

Deliverables

- Test Plan or Strategy Document
- Effort Estimation Document

Test Analysis

This stage answers to the 'What are we testing question?'. The test conditions are understood and accessed not just through the requirements that have been identified at the first stage, but also another related test basis like the product's risks. Other factors that are taken into account while arriving at suitable test conditions are –

- Different levels and depth of testing
- Complexity levels of the product
- Risks associated with the product and the project
- The involvement of the Software Development Life Cycle
- Skillset, knowledge, expertise, and experience of the team
- Availability of the different stakeholders.

Test Design

If the Software Testing Process were answers to a series of questions (which it is), this stage would answer the question – 'How to go about testing the Software?'

The answer, however, depends on a lot of tasks that need to be completed at this point in the process.

These are –

- Working on with the predefined test conditions. This requires breaking down of the test conditions into multiple sub-conditions so that all areas can get their due coverage.
- Identifying and collecting all data related to the test, and using it to set up a test environment conducive to the software.
- Developing metrics to track the requirements and test coverage.

Test Implementation

Now that all the basic structuring work has been done, the next step is to plan how the test structure that has been devised will be implemented.

This means that all test cases are to be arranged according to their priority and a preliminary review is in order to ensure that all test cases are accurate in themselves and in relation to other test cases.

If needed the test cases and test scripts will undergo an additional reworking to work with the larger picture.

Deliverables

- Environment ready with test data set up
- Smoke Test results

Test Execution

When all is said and done, this is where the real action begins. All the planning and management culminates into this – the Execution of the Software Test. This involves a thorough testing of the Software, yes, but also a recording of the test results at every point of the execution process.

So, not only will you be keeping a record of the defects or errors as and when they arise, but you will also be simultaneously tracking your progress with the traceability metrics that have been identified in the earlier stages.

Test Conclusion

This is where the Exit criteria begin by ensuring that all results of the Software Testing Process are duly reported to the concerned stakeholders.

There are different ways of making regular reports, weekly or daily. A consensus is to be arrived at between the stakeholders and the testers, to ensure that parties are up-to-date with which stage is the Software Testing Process at.

Depending on the Project Managers and their awareness of the Software Testing Process, the reports can be intensely technical or written in easily understandable non-technical language for a layman.

Deliverables

- Competed RTM with the execution status
- Test cases updated with results
- Defect Reports

Test Cycle Closure

This last stage is more of *seeing off* of the Software Testing Process. It is where you tick off the checklist and make sure all actions that were started during the process have reached their completion.

This involves making concluding remarks on all actions of the testing process with respect to their execution and/or mitigation.

Also, a revisiting of the entire Software Testing Process as it concludes, will help the team in understanding and reviewing their activities so that lessons can be learned from the testing process and similar mistakes (if any) be avoided in the next Software Testing Cycle the team undertakes.

Deliverables

- Test Closure Report

- Test Metrics

This is a complete guide to the Software Testing Process. While the particulars of the process might vary depending on the specific type of Software Testing Technique that is being used by the team, the process, in general, undergoes all these steps.

Though, the end goal remains the same, i.e., to ensure that the Software has been perfected before it is passed on to the Production Team.

There is no point to having a Software that is fueled with bugs that make it impossible for the end users to use it productively. Therefore, irrespective of how it is undertaken, Software Testing is an important process in the Development of the Software.

Ideas to improve your Software Testing Process

1. Double confirm the requirements from the client. Make sure there are no ambiguities. If there are any get them sorted out at the onset. The entire quality is based on the user requirements; you cannot afford to go wrong in this area.

2. Get involved from the beginning of the development process to get a comprehensive understanding of the AUT which would, in turn, help you identify areas needing more intensive testing

3. During the design phase, you can start working on test cases based on design documents and with the help of the developers. This will mean less time for creating and updating test cases during the actual testing phase.

4. The test plan should be written by a Manager or QA lead. The good test plan will cover deadlines, scope, schedule, risk identifications among other things. Make sure to have it reviewed by the project manager as well for concurrence.

5. Ensure that the quality is maintained right from the beginning irrespective of whether it is development, testing or network support. QA's are like the gatekeepers of quality, they should not let anything pass which is not as per the expected quality.

6. Testing as early as possible and as frequently as possible also helps get more detailed testing done. Do not worry about the number of modules delivered, if you have the bandwidth get the testing done for even a single module.

7. User training should be provided to the QA team for a better understanding of the way the product is actually used by the clients or end users. This will also help them in testing the product better.

8. While creating test cases make sure that all the permutation and combination of different types of data input are covered. Use the well-known principles like boundary value analysis (BVA), Equivalence Partitioning (EP), Decision Table This ensures that the code is tested thoroughly.

9. Always ensure that testing goes in parallel with development. Make sure to test individual modules as and when they are ready for testing instead of waiting for the completion of the entire module.

10. Make use of stubs and drivers to testing smaller modules which requiring input from other modules or are pushing data to other subsystems.

11. Make a distinction between testable and non-testable items in the requirement document and include it as part of the test plan. This will be helpful in having confusion later on.

12. Always ensure that one test case covers only 1 validation point. Though in some cases it may lead to duplication and extra effort for writing the test cases, this is needed for better traceability of the bugs.

13. Test Coverage is an important aspect of ensuring a quality product. Use a traceability matrix to make sure each requirement is mapped to at least one or more test cases. This ensures complete coverage of the system.

14. Ensure that the test case documents are completed, reviewed and signed off by the stakeholders before the actual testing starts. Also, make sure that the test cases document is made available to the rest of the team for understanding and doubt clearance if any.

15. Identify and group test cases based on their importance or priority. This is helpful at the time when only limited test cases can be run during a particular testing cycle. This grouping is also needed when you aim to run or re-run only a particular set based on their priority.

16. You can also have test cases grouped based on functionality. This is useful at the time of regression testing done after a bug fix in any particular module. For example, if there has been a bug fix related to functionality in module A, then you can pick all test cases grouped as module A for the next regression test.

17. Reviews are important to ensure correctness, following of guidelines etc. Ensure that all test cases are reviewed internally within the team (peer review) and once externally also to ensure correctness of the expected results and any other discrepancies.

18. Involve tools to make your life simpler. Tools like Quality Centre, Test Complete, Rally etc. make your life much easier by taking care of a lot of metrics generation and data collection. These tools help you manage your testing projects in a much better and more professional manner.

19. Automation is very important. It reduces manual effort on repetitive tasks. So it is always a good choice to go in for automation of modules or features which are stable and repetitive.

20. Encourage team members to come up with innovative thoughts to make the testing process less time consuming but at the same time more fruitful. Many minds at work is always better than a single one.

21. Always have clear precise and achievable deliverables. Keep a 10% buffer for retesting and other risks. Keeps you and your plan safe and viable.

22. Version control all your artefacts. This is especially important if you are using automation. Make sure the code is version controlled so you are able to create multiple branches and each one should be easy to revert back to in the worst cases if needed.

23. If feasible implement CI/CD. This ensures that each time a build is created the automation suite is run automatically to check if the basic build and its functionalities are in place or not. This helps save a lot of time as well as human error in forgetting to test a particular build.

24. Delegate responsibility to team members for modules or smaller sections. This makes them work harder for ensuring the quality

25. Always look at the software from the eyes of the user as a product and not as a project. The product is most important, think like a consumer when you use the product.

Test Environment and Teams

26. Has the environment set up in such a way that it replicates the production environment as closely as possible?

27. Ensure that all API's, backend systems, user access, admin roles, test data and any other input needed are set up well ahead of the testing cycle. This is to ensure a loss of time in setting up when the testing window in open.

28. Always have a healthy relationship with the development team. This will help in a quick turnaround in case of clarifications.

29. Always ensure to have constant communication with the developer at every stage of the testing process so that everyone is aware of the testing activity in the process.

30. Have at least one person from the testing team join the daily project meetings, so that the testing is aware of the modules coming up for testing in the next release and hence can start preparing for it.

31. During the testing phase, have a daily meeting within the team, to clear any doubts and get additional information where ever needed. This is also helpful in tracking the testing progress.

32. Each team has a role to play in the delivery of a good quality software product. Ensure that each team is assigned clear cut guidelines and areas, avoiding a conflict of interest.

33. Establish a clear and precise communication plan along with SPOC (Single point of contact) list and escalation matrix which will contain a list people or contacts to be used in case of any escalations or in cases where information is needed immediately.

34. It is always advisable to have the team members test a different module each time. The reason is twofold. For one the tester get a better understanding of the complete project instead of a single module and also new bugs can be found if the module is tested by fresh testers each time.

35. Always be prepared for changes and contingencies this also includes the risks. Planning such situations help you avoid panic and face the scenario more gracefully too.

36. Always maintain a risk register and review it frequently. You can also have a time slot set aside with all the stakeholders to review this risk register say weekly. Also, make sure that all risks and mitigation plans are captured accurately.

37. Maintain and update simple metrics as defect found, defects fixed and percentage completion, quality level etc. Send out these metrics to all stakeholders and people involved preferably in a pictorial format, along with the comparison with previous week's data. This will have a much greater impact and visibility than you can imagine. Give it a try.

38. Monitor your production app. And yes for the unfortunate bugs that do find their way to production, make sure they are included as part of the regression test suite with a high priority.

Overall Issues and Reporting

All the information regarding bugs and product development should be maintained in a common Share Point or repository for everyone to access and derive reports when needed.

40. Always have a fresh and open mind when testing software. Have no assumptions. And always refer to the requirements document for doubts and clarifications.

41. The entire product module should be divided into smaller parts to ensure each and every part is tested thoroughly

42. Always ensure to have included the maximum details in the bug report which should include the user credentials, navigations or step to reproduce, expected and actual results, environment, version, logs and screenshots of the error.

43. Always give clear instructions to reproduce an issue. Do not assume that the reader will know the application. Mention each object (button, link, edit box etc.) and action to be performed (click, enter, double click etc.) along with the pages navigated.

44. Discuss and reach a consensus before the starting of the testing on the number of high priority and medium priority bugs can remain open or deferred at the time of movement to production

45. Metrics are important criteria proving the effectiveness of any testing process. Choose appropriate metrics based on the project you are working on. Defect Density, Defect Removal Rate, Testing Efficiency, Test Case Execution Rate etc. are some commonly used software testing metrics.

46. Verbal communications should be topped up by documenting the same in email and shared with relevant stakeholders. This not only creates better visibility but is also helpful in having a chronological record of communications.

47. TDD – Test Driven Development is a newer form of development which takes the base from the test cases. This can also be employed for a better quality of code development.

48. Have a UAT testing at least once before the product is released into production. This will be useful in identifying issues which are related to actual client usage. UAT testing is best performed by the users themselves.

49. Always keep a check on the time taken by the development to fix the bugs. This data is part of an important metrics which is used to calculate the overall efficiency of the development team.

50. Last but not least, trust your instincts. Yes, the QA people sense issues. So just follow your instincts, if you feel a certain area can have issues, mark my words there more chances that issues will be found in that part of the code.

As important as testing and quality assurance is, we should give it the time and resources it needs. This pays off in the long run. These tips will be helpful to you for implementing a good test strategy as well as a quality product.

Happy testing…!!!