American Journal of Bioinformatics
1(1):37 - 39, August 2019

Research Paper

Software Testing - A Test Case Design with Good Understanding.

[1] M.Hari, [2]J.Rajendra Reddy.

[1] Head & [2] Lecturer, Dept. of Computer Science Applications, CSSR & SRRM Degree & PG College, Kamalapuram, Kadapa-516289,A.P., India.

To cite this article: M.Hari, and J.Rajendra Reddy. Software Testing: A test case design with good understanding. American Journal of Bioinformatics, 1(1):37- 39, August 2019.

Email: mudigoti.1986@gmail.com

Received9th June 2019.|Revised: 20th June 2019.|Accepted: 28th June 2019.

Abstract: A Test case is a series of steps designed to test the correctness of software functionality or an important part of an application. Test cases are accepted to be simple and easily understandable. The classic objective of test cases or testing in general is to find defects. A best test case has identifies an error with good probability. When someone reads a test case for the first time, he or she should be able to understand and execute a script without any assistance. Below are certain conventions and guidelines which are to be followed while writing good test cases. Test cases are tricky objective because assessing the quality is multi-dimensional. Still if a tester is able to detect a good number of defects, then test are said to be of good quality. However, if test cases are not exhaustive in nature then the product might fail under conditions that were not covered as a test case. The best, one can do is to introduce different types of testing and ad-hoc testing so that there is enough room to cover any gaps in test cases. One or more test cases should be designed to test one requirement. Effort should be made not to have too many requirements mapped to a single test case. Tracking becomes easier, and is good to have at least one test case per requirement and also yields good test results. Check should be made to verify if the test cases have all the functionalities and requirements covered as per Requirement Traceability Matrix.

Keywords: Testing, Test Case, Quality, Design, defects, understanding.

1. Introduction

A Test case is a series of steps designed to test the correctness of software functionality or an important part of an application. Test cases are accepted to be simple and easily understandable. The classic objective of test cases or testing in general is to find defects. A best test case has identifies an error with good probability. When someone reads a test case for the first time, he or she should be able to understand and execute a script without any assistance. Below are certain conventions and guidelines which are to be followed while writing good test cases. Test cases are tricky objective because assessing the quality is multi-dimensional. Still if a tester is able to detect a good number of defects, then test are said to be of good quality. However, if test cases are not exhaustive in nature then the product might fail under conditions that were not covered as a test case. The best, one can do is to introduce different types of testing and adhoc testing so that there is enough room to cover any gaps in test cases. One or more test cases should be designed to test one requirement. Effort should be made not to have too many requirements mapped to a single test case. Tracking becomes easier, and is good to have at least one test case per requirement and also yields good test results. Check should be made to verify if the test cases have all the functionalities and requirements covered as per Requirement Traceability Matrix.

2. How to Derive the Test Cases

Deriving test cases from use cases can be easily done, as the use case has all sort of information needed for writing test cases? Testing a system using use cases is kind of decision driven testing. We should know the methods and thought processes to recognize test cases from use cases. First of all, we have to draw a model of the use case, based on the use case flow of events and showing the main system user interactions.

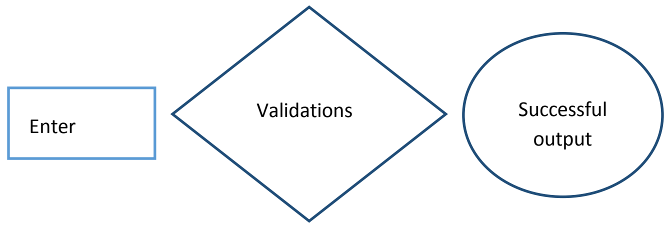

Figure - 1: Derivation of the test cases

2.1. Test case design with good understanding

One needs to have a thorough knowledge of the application and or domain to design good test cases. Test case quality can be attained by writing test cases in conjunction with understanding the business requirements, which can be achieved by going through the Use Cases and all other requirement documents. As a tester, it is important that one makes sure that each and every point mentioned in the business requirement is addressed in test cases and tested with due importance. An easy and effective way to write test case without having missed any requirement is to extract as much test conditions as possible, covering all the main flows and alternate flows specified in a Use Case and additional requirements if any. But there is always an effort that has to be put constantly with time in maintaining test scripts as and when there are some changes or updates being made in requirements. However for better efficiency and test coverage, a tester should be prepared with a wide variety of approaches than only testing based on requirements.

2.2. Correctness of the Test Case

Many times it happens that a test case is wrongly scripted or is not a valid one while executing it. It is important for a tester to know the applicability and correctness of a test case in a given time. The change in requirements may lead to maintain/modify the test cases and sometimes removal the same if they are identified as out of scope. Efforts like thorough review and rework should be made to mitigate the risk of requirements being misinterpreted while scripting the test cases.

2.3. Test Case Design Techniques

The Test Case design techniques are associated with the type of testing performed. A Functional test technique is also called as 'black box' test technique and the common forum is that we are 'doing black box testing'. A functional test technique will help design test cases based on functionality of component or system under test, without necessarily having to understand underlying detail of software design. Consider functionality of system of determine test inputs and expected results.

2.4. Equivalence

Situations (Two or more) are equivalent if they produce essentially the same behavior. If one situation works properly, then it can simply be assumed that the others are correct and need not be tested them all. There are 4 accounts A, B, C and D. Accounts A and B are of same account type and has valid amount for transaction. If transaction is tested between A and D, then there is no need of testing same transaction between B and D.

2.5. Boundary Value

Boundary value testing (positive and negative) test cases are needed. Most of the errors produced to congregate at the boundaries. According to “Transfer Funds” use case, the transaction amount should be less than the available balance of “From Account” and greater than “Zero”. Hence the positive and negative test cases should mainly focus on the boundaries.

2.6. Error Guessing

The test cases are developed by the experienced Test Engineers. For example, where one of the inputs is the date, a test engineer might try July 24, 2000 or 9/9/99.

2.7. Input based Test Techniques

The input data is very important and it can break the system, if the error handling modules are not developed correctly. Once the inputs are recognized, we need to figure out possible set of values, which leads the flow. The variations of input data values (valid and invalid) are adequate for testing. All valid and invalid initial states must be established for the inputs to produce the expected output.

3. Test Case Design Templates

For every identified Test Case a detailed designing should be Prepared in an appropriate format having the following fields.

Test-Case Id: A unique number for a test case.
Test-Case Scenario: Scenario of the test case and the objective of the test case should be described.
Test Data: Test data that has been set up to execute a given test case. Also there should be a mention if this given data is/not reusable.
Pre-req: Pre-requisite/precondition to execute the test case
Post-condition: Post-condition after the test case is successfully run.
Revision History: Revision history to know when and by whom it is created or modified Test Step Number: Step number in incremental fashion and also get the total no. of Steps in a test case
Test Steps (Design): Detailed description of every Step of test execution
Expected Results: Detailed description of the expected result for corresponding test Steps respectively based on the requirements.
Present Results: The result of the action for the given data and how the system reacts for the given data said by the present results.
Pass/Fail: If the Expected and Present results are same then test is Pass otherwise Fail.
A Part from these, one is expected to follow certain test case standards imposed by the organization.

4. Types of Test Cases

4.1. System Test Cases

The system test cases meant to test the system as per the requirements; end to end. Basically to make sure that the application works as per SRS. In system test cases (generally system testing itself), the testers are supposed to act as an end user. The system test must focus on functional groups, rather than identifying the Program units. When it comes to system testing, it is assume that the interfaces between the modules are working fine. (Integration passed). Ideally the test cases are nothing but a union of the functionalities tested in the unit testing and the integration testing. Instead of testing the system inputs outputs through database or external Programs, everything is tested through the system itself. For example, in online shopping application, the catalog and administration screens (program units) would have been independently unit tested and the test results would be verified through the database. In system testing, the tester will mimic as an end user and hence checks the application through its output. There are occasions, where some/many of the integration and unit test cases are repeated in system testing also; especially when the units are tested with test stubs before and not actually tested with other real modules, during system testing those cases will be performed again with real modules/data.

4.2. Path Based Test Cases

To count the number of paths exists in our model and we have to identify the set of test cases needed to exercise the paths through the “Funds Transfer” use case. After identifying the test cases, the feasibility of the test cases is to be analyzed. If the Credit card account is selected as “From Account”, then a warning message will be shown as “Interest charge will be occurred for the transaction”. Hence there are two possible flows,

- Selecting Non-Credit card account as from account.
- Selecting Credit card account as from account.

4.3. Test Case Design Reviews

To ensure that test cases are effective and of good value and the test case review has to be thorough. Often, reviews are ignored owing to time constraints. As Test Cases are considered as a deliverable as far as testing team is concerned, it is important to ensure that test cases are designed to effectively test the product and in such a way that all possible defects come out of the scripts only. Thorough reviews are done during each phase of testing. Let us consider the Test case review Part of It for our ease. Test Case review is not only concentrated. on reviewing the content of a test case, but also in identifying the gaps for missed requirements if any and degree to which test coverage is achieved. It is said to be a good practice for a tester to do a round of self-review before sending the scripts for a peer review or group review. This minimizes the effort spent on reviewing as well as the chances of correction at an Initial stage of test case design. Review comments coming out of self- review, peer review and group review are tracked in tracking tools.. The reviewer opens a defect and assigns it to the designer of the test case. The designer of the script then reworks on the scripts and closes the defect.

Checklist for Test Case review

- Has the correct template been used?
- Have all the expected details to be mentioned in the test case correctly filled in?
- Has the test data set up been done? Has it been included in the test case?
- Have all the positive and negative scenarios, boundary value conditions been covered in test cases?
- Are all scripts free from grammatical and formatting errors?
- Have the Steps been correctly written in appropriate sequence for each test case?
- Have the expected results clearly stated how the system should behave for each Step or action?
- Have all the requirements been addressed in test cases?
- Has the navigation correctly documented in test Steps without any discrepancies?
- Has the naming convention used for Test Scenarios and Test Cases as per the standards?

5. Conclusion

Software testing is the process used to assess the quality of software and it may be viewed as an important part of the software quality assurance. A mistake made by the programmer, the results in a error in the software source code. If error is executed, in some situations the system will produce false results. A problem with software testing is that testing all combinations of inputs and pre conditions are not feasible when testing anything other than a simple product. All test cases should be traceable to all requirements and test case should be planned ahead of execution. Certain bugs easier to find in testing and most of the defects are uncovered at the initial stages by using reviews. A high standard of finding undiscovered errors are possible through successful test case design. Test plans and test cases were developed in the analysis phase and they are revised. A successful test case design is discussions and experience work.

References

1. Lee Copeland. A practitioner's guide to software test design. First Edition, Artech house, incorporated.

2. Roger.S.Pressman. Software engineering- a practitioner’s approach. seventh edition, Mc Graw hills international edition.

3. Paul.C.Jorgensen . Software testing - a craftsman's approach. 4th edition, Pearson Education.

4. C.Kaner. Testing computer software.2nd edition, Wiley publications.

5. http://www.onestoptesting.com/test-cases/designing.asp

6. http://www.onestoptesting.com/test-cases/template.asp

7. http://www.onestoptesting.com/test-cases/good-test-case.asp

8. https://www.utest.com/articles/how-to-design-test-cases-for-testing-part-1-equivalence-partitioning

9. http://www.softwaretestinggenius.com/how-to-design-a-good-test-case-for-performance-testing