Generating Test Cases for E-Commerce Systems

Khalid Alzubi
Albalqa Applied University, Salt, Jordan

Abstract

Software testing is any activity aimed at evaluating a capability of a program or system to determine that it meets its required results and detect defects.

In this paper main testing techniques are shortly described and their classification is outlined. It focuses on Purpose Software Testing Techniques and examines different examples for each and every testing technique.

Keywords: Software Testing, Correctness Testing, Performance Testing, Reliability Testing, Security Testing

1. Introduction

Software Testing is the process of executing a program or system for finding any error or defect. Software Testing Strategy helps to convert test case designs into well- planned execution steps that will result in the construction of successful software. The primary goal of test cases is to derive set of tests that have the highest probability of uncovering the errors.

Software testing is a destructive process of trying to find the errors. The main purpose of testing is quality assurance, reliability estimation, validation or verification.

Despite its limitations, testing is an important part in software development. It is broadly deployed in every phase in the software development cycle. Typically, more than 50% percent of the development time is spent in testing [9].

2. TESTING

Software testing is an activity used for finding errors in software. It also verifies and validates whether the program is working correctly and smoothly with no bugs. It analyzes the software system to find such bugs. Also Software testing confirms that either the software is working according to the requirement specifications or not. Software testing includes a number of steps which is designed to make sure that computer code does what it was designed to do [10].

There is a dire need for Software testing for if we fail to deliver a reliable, good and error free software solution, our project fails and we may lose clients. Thus, in order to guarantee proper software solution, we go for testing. We test for any problem or error in the system, which can make software unusable by the client. We make software testers test the system and help in finding out the bugs in the system to fix them on time [2]. Software testing is a process of measuring the quality of the developed software. It is also a process of uncovering bugs in a program and makes it active. It is a useful process for executing the program.

In order to ensure the software quality, one conducts software testing in every phase of the software development process. A complete software testing should cover the entire life cycle of software product. [1]

2.1 Software Testing Objectives:

Main goals of testing can be quality assurance, reliability estimation, validation or verification. Testing includes other objectives:

- Testing is the process of executing a program to find errors.
- A good test case is one for finding an undiscovered error with a high probability.
- A successful test is one that discovers an undiscovered error.
- The software that works better can be tested more efficiently.
- Testing is the process that identifies the correctness and completeness of the software.

2.2 Software Testability

The test can be good and efficient depending on the software itself, and here a checklist for the software testability is needed

- Operability – when it works better, it can be tested more efficiently.
- Observability - what you test is what you see.
- Controllability – controlling the software better makes testing more automated and optimized.
- Decomposability - problems can be isolated and retested intelligently more quickly by controlling the scope of testing.
- Simplicity - we can test quickly if the test is less complex.
- Stability – with few changes, disrupting testing is minimized.
- Understandability - the more information known, the smarter the testing.

3. Software testing classification

Testing methods and testing techniques are different and serve multiple purposes in different life cycle phases; they are classified into four groups.

By purpose Classification, software testing can be divided into correctness testing (White-box testing, Black-box testing, Gray-box testing), performance testing, reliability testing and security test.

By life-cycle phase Classification, software testing can be classified into the following categories: requirements phase testing, design phase testing, program phase testing, evaluating test results, installation phase testing, acceptance testing and maintenance testing.

By scope Classification, software testing can be categorized as follows: unit testing, component testing, integration testing, and system testing.

By Fault Based Methods Classification, which include Error Based Testing, Fault seeding, mutation testing, and fault injection, among others.

4. Test cases

4.1 Test case definition

IEEE Standard 610 (1990) defines *test case* as follows:

"(1) a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

"(2) (IEEE Std 829-1983) Documentation specifying inputs, predicted results, and a set of execution conditions for a test item."[1]

Then a test case is a question is put to the program. The test goal is to gain information as to whether the program will pass or fail the test.

When running tests and test cases, try to achieve some points:

- Find bugs and report.
- Fix these bugs.

4.2 Writing a good test case

Test cases can be "good" in different ways. There's no simple style or prescription for writing "good" test cases.

According to the architecture of the system, 90% of the system complexity is in the module analyzing. In this case, it is clear that the test-case selection and test method should be different.

Testing may not cover some of the important sides of the application or system, for instance, by selecting only the expected interactions when testing an application or testing only some subset of the specified functions. In most cases, it is more important to focus on determining the technical correctness of a system. It is needed to prioritize all requirements and the functions that are likely to contain critical problems. Most existing applications are too complex to test in every possible way, through all possible paths and states. Prioritizing the paths and scenarios that are tested first is useful, timesaving lesson for a test team, especially when resources are limited.

Testing activities can fail in many ways; however, most problems can be prevented with the following practices:

• form a suitable test team with the appropriate means for performing the tests at hand.

• make testing an integral part of software development.

• employ management changing processes.

• ensure requirement traceability to and from tests.

• automate test specification and execution.

• Testability design.

Given the complexity of current and expected software and communications systems, it is expected that software testing will become even more complicated. Then, even more strong tools and methodologies will emerge over time. Manual testing is becoming a less viable alternative, and integration with the overall design processes and tools will prove necessary to keep pace in testing these complex current and future systems.

5. Purpose Software Testing Techniques

The most prevalent techniques of software testing are classified by purpose. Fig. 1 Represent software testing techniques which are classified by purpose

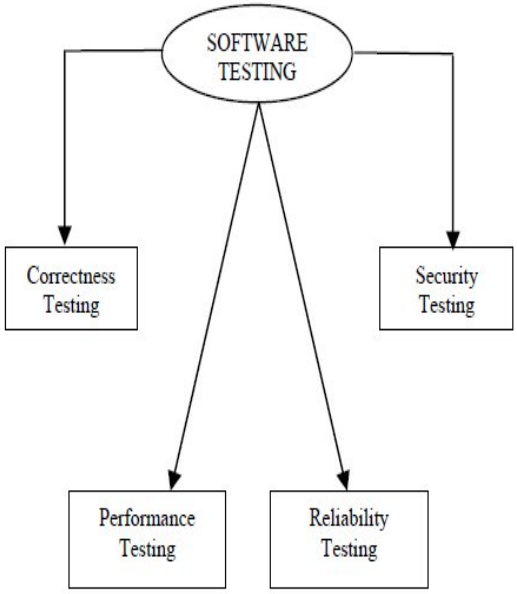

Fig.1 software testing techniques [5]

a. Correctness Testing

Correctness is the minimum requirement of software, the essential purpose of testing. Correctness testing will need some type of oracle, to tell the right behavior from the wrong one. The tester may or

may not know the inside details of the software module under test, e.g. control flow, data flow, etc. Therefore, either a white-box point of view or black-box point of view can be taken in testing software. We must note that the black-box and white-box ideas are not limited in correctness testing only [9].

Correctness Test case example

Testing for an input box accept numbers from 1 to 1000 then there is no use in writing thousand test cases for all 1000 valid input numbers plus other test cases for invalid data.

Using equivalence partitioning method, test cases can be divided into three sets of input data called as classes. Each test case is a representative of respective class.

So test cases were divided into three equivalence classes of some valid and invalid inputs.
1) One input data class with all valid inputs. Pick a single value from range 1 to 1000 as a valid test case. If you select other values between 1 and 1000 then result is going to be the same. So, one test case for valid input data should be sufficient.

2) Input data class with all values below lower limit. I.e. any value below 1, as an invalid input data test case.
3) Input data with any value greater than 1000 to represent third invalid input class.
So using equivalence partitioning means that you have categorized all possible test cases into three classes. Test cases with other values from any class should give you the same result [14].

b. **Performance Testing**

Not all software systems have specifications on performance explicitly. But every system will have implicit performance requirements. The software should not take infinite time or infinite resource to execute. "Performance bugs" are sometimes used to refer to those design problems in software that cause the system performance to degrade.

Performance has always been a great concern and a driving force of computer evolution. Performance evaluation of a software system usually includes: resource usage, throughput, and response time and queue lengths detailing the average or maximum number of tasks waiting to be serviced by selected resources. Typical resources that need to be considered include network bandwidth requirements, CPU cycles, disk space, disk access operations, and memory usage. The goal of performance testing can be performance bottleneck identification, performance comparison and evaluation, etc [9].
Test case generation requirements:

- Performance test cases should cover the most used scenarios of end-users' behavior in the system (website, application) to emulate load.
- Include to test cases steps with user actions which performance you want to measure.
- Separate performance test cases by logical parts of the system which you are going to test.
- Number of performance test cases should be as small as possible, but enough to cover important points (typical number is 5-15).
- Number of steps in performance test case should be as small as possible, but

enough to repeat usual user steps (typical number is 3-10).

- If system has some caching rules on the server side test case should be written in a way that allows avoiding influence of caching mechanism on performance testing results.
- Test case may describe real user's behavior [16].

Test case example

Writing a test case for performance testing is basically writing a simple Requirement Specification for a piece of software .Just as with any specification, it should be unambiguous and as complete as possible.

Test Case: Modify Item				
Description: This test case simulates one of the actions a stock adjuster would perform each day. The user will search for item by Item ID, and then modify the item description.				
Data Requirements: {Username} – User must have update privileges. User name must be unique (as application does not allow simultaneous logins). {Password} – must be valid for given {Username} {ItemID} – any item that is currently in stock may be used. Items should be selected at random. Use the following SQL query: "select item_id from items where quantity > 0". Note: if two users open the same item for modification and one saves the item. The second user will not be able to save their changes. {ItemDescription} – The new item description should be the same as the old item description, except with "modified" added to the end.				
Step Number	**Step Description**	**Expected Result**	**Transaction Name**	**User Think Time**
01	Invoke application from desktop icon. Log in with **{Username}** and **{Password}**	Main menu screen is displayed	user_login	5
02	Select item search from menu.	Search screen is displayed	select_search	2
03	Enter **{ItemID}** in Exact Find field. Press Search button.	Item properties screen is displayed.	search_by_item_id	10
04	Press Edit button	Item for specified **{ItemID}** is displayed in edit mode.	press_edit	1
05	Modify the **{ItemDescription}** in the Description field. Press Save button.	Item properties screen is displayed	modify_description	20
06	Press Main Menu button	Main menu screen is displayed.	return_to_main_menu	5
	Return to step 02 and repeat.			

Fig.2 Performance example [15].

c. **Reliability Testing**

'Reliability Testing' is very important, as it pinpoints all the failures of a system and removes them before the system is deployed. Reliability testing is related to many aspects of software in which testing process is included; this testing process is an effective sampling method to measure software reliability. Estimation model is prepared in reliability testing which is used to analyze the data to estimate the present and predict future reliability of software [5].

Based on reliability information, the risk of using software can also be evaluated.

d. **Security Testing**

Security Testing makes sure that only the authorized person can access the program and only the authorized personnel can access the functions available to their security level. Security testing is very helpful for the tester for finding and fixing problems. It ensures that the system will run for a long time without any major problem. It also ensures that the systems used by any organization are secured against any unauthorized attack [10].

With the development of the Internet, software security problems are becoming even more serious.

Many critical software applications have integrated security measures against malicious attacks. Simulated security attacks can be performed to find weakness points.

Test case example

Before beginning to write a test case: 1) It is important to segregate based on Roles (something like Admin, Manager,

and Supervisor etc.)
2) Delve into the negative scenario for a particular event initially before taking up the positive scenarios. This will ensure continuity of the test cases and will greatly help [13]. The table below shows an example for security testing.

N0.	Action	Test steps	Pass/fail
1	Invoke the application by typing the URL "http://....;	The browser should be invoked and the application login page should appear.	
	Verify the login security for the Project Lead.		
2	Login with login name as "abhilash" and password as "password56".	The user should be logged in and be directed to the Home page.	
3	Verify the menu structure on the Home page.	The home page should contain the following menu structure. – Projects – Tasks – Dashboard – Reports – Skills	
4	Verify the menu dropdown for the "Projects" menu.	The "Projects" menu should contain the following menu items. – Create Task – Create Build – Create Module	
	Like this you will need to cover the other menus too.		
	End of verification for the Project Lead.		

Table1 security example[13].

6. CONCLUSION

Software testing is an important technique for the improvement and measurement of a software system quality.
In this paper we proposed that Software testing can be very costly. A good way to cut down time and cost is to generate good test cases.
A few cases and examples are considered in this study and they are only used to provide a clear explanation regarding testing techniques.

REFERENCES

[1] Cem Kaner, J. D. (2003). What Is a Good Test Case? *Proceedings of Florida Institute of*

Technology Department of Computer Sciences STAR East.

[2] Gandhi, G., & Garg, S. Implementing Software Testing Model Approach for Efficient Bug Finding with Yin-Yang Testing Theory on Java Application.

[3] Jovanovic, I. (2008). Software testing methods and techniques. *IM Jovanovic is with the Inzenjering, Mat*, *26*.

[4] Khan, M. E. (2011). Different Approaches to White Box Testing Technique for Finding Errors. *International Journal of Software Engineering and Its Applications*, *5*(3).

[5] Khan, M. E. (2010). Different forms of software testing techniques for finding errors. *International Journal of Computer Science Issues, May*, *7*(3), 11-16.

[6] Luo, L. (2001). Software testing techniques. *Institute for software research international Carnegie mellon university Pittsburgh, PA*, *15232*(1-19), 19

[7] Nidhra, S., & Dondeti, J. (2012). BLACK BOX AND WHITE BOX TESTING TECHNIQUES–ALiterature REVIEW. *International Journal of Embedded Systems and Applications (IJESA)*, *2*(2), 29-50.

[8] Nirpal, P. B., & Kale, K. V. (2011). A Brief Overview Of Software Testing Metrics. *International Journal on Computer Science and Engineering (IJCSE)*,*3*(1).

[9] Pan, J. (1999). Software testing. *Retrieved September*, *2*, 2013.

[10] Pardeshi, S. N. Study of Testing Strategies and availableTools. *International Journal of Scientific and Research Publications, ISSN*, 2250-3153.

[11] Thakare, S., Chavan, S., & Chawan, P. M. Software Testing Strategies and Techniques..

[12] Williams, L. (2004). Testing overview and black-box testing techniques. *Alamat situs: http://www. agile. csc. ncsu. edu/ SEMaterials/BlackBox. Pdf*.

[13]https://eccentricabhi.wordpress.com/2009/07/15/writing-test-cases-for-security-test-role-based/ [26/12/2014]

[14] http://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/ [26/12/2014]

[15] http://www.myloadtest.com/how-to-write-a-performance-test-case/ [26/12/2014]

[16] http://loadtestcrew.com/2013/02/performance-test-cases/ [26/12/2014]