Software Testing, Mythology & Methodologies

Sonali Waje¹, Vandana Gaikwad², Pranchal Chaudhari³
¹,²B.E. Information Technology, ³B.E. Computer Engineering

Abstract - It is generally believed that testing phases of software development consume 30% to 50% of the entire software development time and resources. The cost and time of testing should get reduced to increase the productivity of the software development process. To achieve these goals, different levels and tools of automated testing used rather than manual testing. There are most common myths about the software testing. In this paper we are trying to reduce the impact of those myths on software testing. The paper is elaborating mythologies and different methodologies of software testing and the testing tools being used.

Keywords - Automated testing, Manual testing, Methodology, Mythology, Software testing, Testing tools.

I. INTRODUCTION

Software Development lifecycle

(SDLC) is a software process model which follows sequential way of building product.

Software testing is an integral and important phase of the software development process. This part of process ensures that defects are recognized as soon as possible.

As per the ANSI/IEEE 1059 standard, Testing can be defined as a process of analyzing a software item to detect the differences between existing and required conditions (that is defects/ errors/ bugs) and to evaluate the features of the software item.

We introduced below most popular and common myths about software testing.

II. SOFTWARE TESTING MYTHOLOGY

Common myths are:

1: Testing is too expensive.

Reality: There is a saying, pay less for testing during software development or pay more for maintenance or correction later. Early testing saves both time & cost in many aspects however, reducing cost without testing may result in the improper design of a software application depicting the product useless.

2: Testing is time consuming.

Reality: During SDLC phases testing is never a time consuming process. However, Finding and fixing the error which is identified during proper testing is time consuming but productive activity.

3: Testing cannot be started if product is not fully developed.

Reality: Testing depends on source code but reviewing requirements and developing test cases is independent from the developed code. Iterative or incremental approach as a development life cycle model may reduce the dependency of testing on fully developed software.

4: Complete Testing is Possible.

Reality: It becomes an issue when a client or tester thinks that complete testing is possible. It is possible that all the paths have been tested by the team but occurrence of complete testing is never possible. There might be some scenarios that will never be executed by the test team or the client during software development life cycle and may be executed once the project has been deployed.

5: If software is tested then it must be bug free.

Reality: This is a very common myth which Clients, Project Managers & the management team believe in. No one say with absolute certainty that a software application is completely bug free even if a tester with superb testing skills has tested the application.

6: Missed defects are due to Testers.

Reality: It is not a right approach to blame testers for bugs that remain in the application even after testing has been performed. This myth relates to the Time, Cost & Requirements changing Constraints. However, the test strategy may result in bugs being missed by the testing team.

7: Testers should be responsible for quality of product.

Reality: It is a very common misinterpretation that only testers or the testing team should be responsible for product quality. Tester’s responsibilities include the identification of bugs to the stakeholders and then it is their decision whether they will fix the bug or release the software without fixing it. Releasing the software at the time puts more pressure on the testers as they will be blamed for any error.

673
8: Test automation must be used wherever it is possible to use it and to reduce time.

Reality: Yes, it is true that Test Automation reduces the testing time but it is not possible to start Test Automation at any time during Software development. Test Automation should be started when the software has been manually tested and is stable to some extent. Moreover, Test Automation cannot be used if requirements keep changing. In this situation manual testing is preferable.

9: Any one can test software application.

Reality: People outside the IT industry think that anyone can test the software and testing is not a creative job. However a tester knows very well that this is myth. Thinking about alternatives scenarios, try to crash software with the intent to explore potential bugs is not possible for the person who developed it.

10: A tester's task is only to find bugs.

Reality: Finding bugs in the software is the task of testers but at the same time they are domain experts of particular software. Developers are only responsible for specific component or area that is assigned to them but testers must understand the overall working of the software, what the dependencies are & what the impacts of one module on another module are.

III. SOFTWARE TESTING TYPES

This topic describes the different types of testing which may be used to test Software during SDLC.

a) Manual testing

This type includes the testing of the software manually i.e. without using any automated tool or script. In this type the tester takes over the role of an end user and test the Software to identify any unexpected behavior or bugs in the software. There are different stages for manual testing like unit testing, Integration testing, System testing & User Acceptance testing. Testers uses test plan, test cases or test scenarios to test the Software to ensure the completeness of testing. Manual testing includes exploratory testing as testers explore the software to identify errors in it.

b) Automation testing

Automation testing which is also known as Test Automation, is when the tester writes scripts & uses software to test the software. This process involves automation of manual process. Automation Testing is used to re-run test scenarios that were performed manually, fast & repeatedly.

Apart from the regression testing, an Automation testing is used to test the application from load, performance & stress point of view. It increases the test coverage, improves accuracy and saves time & money as compared to manual testing. Automation is done by using supportive computer language like virtual basic scripting and an automated software application.

Figure 1: Automation Testing Cycle

Automated Software testing tools

HP Quick Test Professional [1]
Selenium [2]
IBM Rational Functional Tester [3]
Silk Test [4]
Test Complete
Testing Anywhere [5]
Win Runner [6]
Load Runner [7]
Visual Studio Test Professional [8]
WATIR [9]

IV. SOFTWARE TESTING METHODOLOGY

There are different methods which can be used for Software testing. This paper briefly describes only those common and popular methods.

a) Black Box Testing

The technique of testing without having any knowledge of the core workings of the application is Black Box testing. The tester is unaware to the system architecture and does not have access to the source code.

Typically, when performing black box test, a tester will interact with system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
b) White Box Testing

White box testing is the detailed analysis of internal logic and structure of code. White box testing is also called glass testing or open box testing. In order to perform this white box testing on application, the tester needs to acquire knowledge of the internal working of the code.

The tester needs to have a look inside source code and find out which unit/chunk of the code is behaving inappropriately.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Well suited &amp; efficient for large code segments.</td>
<td>• Limited coverage since only selected number of test scenarios is actually performed.</td>
</tr>
<tr>
<td>• Code Access not required.</td>
<td>• Inefficient testing, due to the fact that a tester only has limited knowledge about an application.</td>
</tr>
<tr>
<td>• Clearly separates user's perceptions from developer's perspective through visibly defined roles.</td>
<td>• Blind coverage, since tester cannot target specific code segments or error prone areas.</td>
</tr>
<tr>
<td>• Large numbers of fairly skilled testers can test the application with no knowledge of implementation and programming language or operating systems.</td>
<td>• The test cases are complicated to design.</td>
</tr>
</tbody>
</table>

C) Grey Box Testing

Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application. In software testing, the phrase the more you know the better carries a lot of weight when testing an application.

Mastering the domain of system always gives the tester an edge over someone with limited domain knowledge. Unlike black box testing, where tester only tests the application's user interface, in grey box testing, the tester has right to use design documents and the database. Having this knowledge, the tester is capable to better prepare test data and test scenarios when making the test plan.
Table 3
Grey Box Testing

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Offers combined advantages of black box and white box testing wherever possible.</td>
<td>• Since the access to the source code is not available; the ability to go over the code and test coverage is limited.</td>
</tr>
<tr>
<td>• Grey box testers don't rely on source code; instead they rely on interface</td>
<td>• The tests can be unnecessary if the software designer has already run a test case.</td>
</tr>
<tr>
<td>• Testing every possible input stream is impractical because it would take an unreasonable amount of time; therefore, many program paths will go untested.</td>
<td></td>
</tr>
</tbody>
</table>

Table 4
Black Box vs. Grey Box vs. White Box

<table>
<thead>
<tr>
<th>SN</th>
<th>Black Box Testing</th>
<th>Grey Box Testing</th>
<th>White Box Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The internal working of an application are not required to be known</td>
<td>Somewhat knowledge of internal workings are known</td>
<td>Tester has full knowledge of internal workings of the application</td>
</tr>
<tr>
<td>2.</td>
<td>Also known as closed box testing, data driven testing &amp; functional testing</td>
<td>Another term for grey box testing is luminous testing as the tester has limited knowledge of insides of an application</td>
<td>Also known as clear box testing, structural testing or code based testing</td>
</tr>
</tbody>
</table>

3. Performed by end users & also by testers and developers | Performed by end users & also by testers & developers | Normally done by testers and developers |
4. Testing is based on external expectations - Internal behavior of application is unknown | Testing is done on basis of high level database Diagrams & data flow diagrams | Internal workings are fully known & the tester can design test data accordingly |
5. This is the least time consuming & extensive | Partly time consuming & extensive | The most extensive & time consuming type of testing |
6. Not suited to algorithm testing | Not suited to algorithm testing | Suited to algorithm testing |
7. This can only be done by trial & error method | If known, data domains & internal boundaries can be tested | Data domains & internal boundaries can be better tested |

V. CONCLUSION

The levels and tools of automated testing is used in order to reduce the cost and time of testing so that it will increase the productivity of the software development process. There are most common myths about the software testing. It is very important to reduce the impact of those myths on software testing. The paper we are presenting is elaborating mythologies and methodologies of software testing and the testing tools being used. Future different works will be dedicated to improving this system. We are also planning to add mailing and message systems into it. We are planning to add graphic statistics tools with software testing tools into it.
REFERENCES


