A Survey of Effective and Efficient Software Testing

Akalanka Mailewa and Jayantha Herath
Dept. of Computer Science
St. Cloud State University
St. Cloud, Minnesota 56301
dmakalanka@gmail.com
herath@stcloudstate.edu

Susantha Herath
Dept. of Information Systems
St. Cloud State University
St. Cloud, Minnesota 56301
sherath@stcloudstate.edu

Abstract

The development of large software systems is a complex and error prone process. Errors may occur at any stage of software development. These errors, sometimes referred to as bugs, can cause great losses in terms of both time and money if not identified and removed as early as possible. Software testing plays a major role in software development life cycle (SDLC). It ceases the propagation of errors and thus decreases costs. But software testing itself can be costly. Research point out more than 50% of the total cost of development is devoted to software testing. However, testing is essential to ensure that the software is capable of doing what it was designed to do.

Testing becomes expensive since it takes much time. This is because of the need to test many combinations of its functions, integrity, performance and so on, which can be called as test cases. A software company's goal would be to reduce the testing time, so that they can save money and deliver the product much faster to the customer. Testing time can be reduced in two main ways, first by reducing number of test cases and second by automating repeatedly tested areas.

This paper will discuss fundamentals of testing such as importance and difference of verification and validation, recent advances in agile software testing, testing throughout the SDLC and testing methods, levels and types. Finally as a conclusion it will also shed some light on combinatorial testing to reduce number of test cases and test automation to reduce manual testing time as the solutions to long testing time problem.

1 Introduction

Software testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product meets the business and technical requirements, works as expected and can be implemented with the same characteristics [1], in other words testing means comparing the actual outcome and expected outcome.

Effectiveness and Efficiency plays a major role in software testing. Test effectiveness is the relative ability of testing strategy to find bugs in the software. Test efficiency is the relative cost of finding a bug in the software under test. One of the major problems in software industry is testing time. Time means money and hence the entire process is costly. Therefore to reduce testing cost it is necessary to reduce manual repeatedly testing time and number of test cases generated for particular system.

Testing is a never ending process, i.e. it cannot be said that the particular software is 100% error free. Therefore we have to define when to start and when to stop testing. In order to maintain and manage entire testing process it is always better to have a Software Testing Life Cycle (STLC) inside the Software Development Life Cycle (SDLC).

1.1 Software verification and validation

Both verification - a form of static testing - and validation - a form of dynamic testing - play a major role in software testing.

1.1.1 Verification

Verification makes sure that the product is designed to deliver all functionality to the customer. Verification is done at the starting of the development process. It includes reviews and meetings, walkthroughs, inspection, etc. to evaluate documents, plans, code, requirements and specifications. It answers the questions like: Am I building the product right? Am I accessing the data right (in the right place; in the right way)? [2].

According to the Capability Maturity Model (CMMI-SW v1.1) we can also define verification as the process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. [IEEE-STD-610].

1.1.2 Validation

Validation is about determining if the system complies with the requirements and performs functions for which it is intended for and meets the organization’s goals and user needs. Validation is done at the end of the development process and takes place after verifications are completed. It answers the question like: Am I building the right product? Am I accessing the right data (in terms of the data required to satisfy the requirement). Validation performed after a work product is produced against established criteria ensuring that the product integrates correctly into the environment also determination of

correctness of the final software product by a development project with respect to the user needs and requirements [2].

•According to the Capability Maturity Model (CMMI-SW v1.1) we can also define validation as the process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements. [IEEE-STD-610].

1.1.3 Summary

Criteria	Verification	Validation
Definition	The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase.	The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements.
Objective	To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements.	To ensure that the product actually meets the user's needs, and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfills its intended use when placed in its intended environment.
Question	Are we building the product right?	Are we building the right product?
Evaluation Items	Plans, Requirement Specs, Design Specs, Code, Test Cases	The actual product/software.
Activities	• Reviews • Walkthroughs • Inspections	• Testing

Table 1: The difference between verification and validation

1.2 Some important aspects of testing

Here the paper focuses on who is going to test the product and most important two things, when to start testing and when to stop testing. As mentioned repeatedly testing is costly, if an organization tests every aspect of software with no time limit it is a waste of time and thus a waste of money. Therefore deciding when to start and when to stop is very important in software testing.

1.2.1 Who does testing?

While large IT companies have their own test team, their smaller counterparts may have only a single tester. Some organizations hire software testers from outside or outsource testing process. The testers' responsibility is to evaluate the software with the given requirements. Sometimes developers and testers may conduct unit testing where they may

separately test each individual component. Unit testing can take place before the whole system has been developed. Following people are involved in testing of a system [3].

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

Different companies have difference designations for those who test the software such as Software Tester, Software Quality Assurance Engineer, and QA Analyst etc. [3]. This is on the basis of their experience and knowledge.

1.2.2 When to Start Testing?

Testing may take place at any stage of Software Development Life Cycle (SDLC), but it is not possible to test the software or certain parts of it at any time during its cycle. An early start to testing reduces the cost and time of rework. In SDLC, testing can start from the Requirements Gathering phase and lasts till the deployment of the software. However it also depends on the development model that is being used. For example in Waterfall model formal testing is conducted in the Testing phase, but in incremental model, testing is performed at the end of every increment/iteration. In the end the whole application is tested [3]. Testing is done in different forms at every phase of SDLC. Following situations can also be considered as testing:

- Analysis and verifications of requirements during requirement gathering phase.
- Reviewing the design in the design stage.
- Testing performed by a developer on completion of each section of code.
- Indirect testing by the end user after implementation.

1.2.3 When to Stop Testing?

This is one of the most difficult questions for a test engineer. Following are few of the common tips on deciding when to stop testing [4]:

- All the high priority bugs are fixed.
- The rate at which bugs are found is too small.
- The testing budget is exhausted.
- The project duration is completed.
- The risk in the project is under acceptable limit.

Practically, the decision to stop testing is based on the level of the risk acceptable to the management. Though we can never assume that 100 % testing has been done, we can minimize the risk of shipping the product to client with X testing done. The risk can be measured by Risk analysis but for small duration / low budget / low resources project, risk can be deduced by simply: Measuring test coverage and/or Number of test cycles and/or Number of high priority bugs [4].

2 Manual testing and automated testing

The difference between manual testing and automated testing is discussed in this section of the paper. Executing the test cases manually without any tool support is known as manual testing. Taking tool support and executing the test cases by using automation tool(s) is known as automated testing. Following two sections will discuss the differences in detail with their advantages and disadvantages.

2.1 Manual testing

Manual testing is the process through which software developers run tests manually, comparing program expectations and actual outcomes in order to find software defects. These manual tests are no more than the tester using the program as an end user would, and then determining whether or not the program acts appropriately. Manual testing is a good fit for smaller projects as well as companies without significant financial resources [5].

2.1.1 Advantages of Manual Testing

1. Software automation tools are expensive. Short-term cost is much lower with manual testing.

2. Automated tests don't necessarily act as human beings. Hence some of the issues the end user might encounter might be overlooked in an automated test. Manual testing, on the other hand, allows the developing program to be used as it would be by an end user. A human user might handle the program in a peculiar way which might give rise to errors that are more likely to be skipped in an automated test but caught with manual testing.

3. Manual tests are flexible than automated tools. Automated tools run according to a set of rules. Test cases are set up and programmed into the automated tool, and then the tests are run. When a change occurs in the project the whole process might have to be repeated. But with manual testing this can be easily incorporated into testing routine.

4. With manual testing, you can quickly see the results. Automatic tests take more time to set up, and sometimes the whole set of tests have to be run before the results are output. Therefore manual tests allow you to test ideas quickly and easily.

2.1.2 Disadvantages of Manual Testing

1. Certain tests are difficult to be done manually, e.g. low level interface regression testing is extremely difficult to be performed manually. As a result, it is prone to mistakes and overseen when done by manually. Automated testing, once set up, is much better equipped to find errors for this kind of testing.

2. Manual testing can be repetitive and boring – no one wants to keep filling out the same forms time after time. As a result, many testers have a hard time staying engaged in this process, and errors are more likely to occur.

3. With automated testing, if you add data to the program, you can rerun all of the required tests instantly because the tests are already set up. This isn't the case with manual testing. If there is any change to the software, you have to run the tests again by hand. This is valuable time lost.

2.2 Automated testing

Automated testing is the process through which automated testing tools run tests that repeat predefined actions, comparing a developing program's expected and actual outcomes. If the program expectations and outcomes align, your project is behaving as it should, and you are likely bug free. If the two don't align, however, there is an issue that needs to be addressed. You'll have to take a look at your code, alter it, and continue to run tests until the actual and expected outcomes align [5].

2.2.1 Advantages of Automated Testing

1. Runs tests quickly and effectively: While the initial setup of automated test cases may take a while, once you've automated your tests, you're good to go. You can reuse tests, which is good news for those of you running regressions on constantly changing code. You won't have to continuously fill out the same information or remember to run certain tests. Everything is done for you automatically.

2. Can be cost effective on the long-run: While automation tools can be expensive short-term, but they save you money in the long-term. They not only do more than a human can in a given amount of time, they also find defects quicker. This allows your team to react more quickly, saving you both precious time and money.

3. Filling out the same forms time after time can be frustrating. But test automation solves this problem. The process of setting up test cases takes coding and thought, which keeps your best technical minds involved and committed to the process.

4. Everyone can see results: When one person is doing manual testing, the rest of the team can't see the results of the tests being run. With automated tests, however, people can sign into the testing system and see the results. This allows for greater team collaboration and a better final product.

5. Automated testing is more suitable when the project is large (i.e. no. of inputs/ configurations are high), there are many system users, or when filling out forms [5].

2.2.2 Disadvantages of Automated Testing

1. The automation tools can be expensive to purchase. As a result, it is important to only use the ones that will give you full, or as close to full coverage, as you can find.

2. While the automation process cuts down on the time it takes to test everything by hand, automated testing is still a time intensive process. A considerable amount of time goes into developing the automated tests and letting them run.

3. Tools have limitations. While automated tests will detect most bugs in your system, there are limitations. For example, the automated tools cannot test for visual considerations like image color or font size. Changes in these can only be detected by manual testing, which means that not all testing can be done with automatic tools.

2.3 Performance Comparison

Following table shows the brief but comprehensive explanation of deference between manual testing and automated testing.

Manual Testing	Automation Testing
Time consuming and tedious: Since test cases are executed by human resources it is slow and tedious.	Fast execution: Automation runs test cases significantly faster than human testers.
Huge investment in human resources: Test cases need to be executed manually so more testers are required in manual testing.	Less investment in human resources: Test cases are executed by using automation tool(s) so lesser number of testers are required in automation testing.
Less reliable: Manual testing is less reliable as tests may not be performed with precision each time because of human errors.	More reliable: Automation tests perform precisely same operation each time they are run.
Non-programmable: No programming can be done to write sophisticated tests which fetch hidden information.	Programmable: Testers can program sophisticated tests to bring out hidden information.
The results are late: programmers cannot see the result until the tester note down, type, print and copy the results.	Quick results: the programmers can see the result real-time in a networked environment.
Might catch more of the errors occurring due to human nature as the testers are human beings themselves.	Might skip errors occurring due to human nature as software cannot 'think' as human beings.
Can be cheaper for small software than automated testing.	Can be costly for smaller projects but cost-effective for larger projects.
Much better at visual testing.	Weak at visual testing. Software do not think like humans and thus it cannot decide whether a GUI is attractive, distractive or dull.

Table 2: The difference between manual testing and automation testing

3 Agile Software Testing

The word agile more commonly suggests a focused yet rapidly iterative software process adhering to principles that were first outlined in the Agile Manifesto first laid down in 2001. The manifesto aimed to promote a more efficient way of developing computer programs and IT systems through a collaborative system of various teams. Today, the agile method has become widely accepted as an effective approach to project management within the mainstream software development and testing community [6]. Agile testing refers to the practice of testing software for bugs or performance issues within the context of an agile workflow [6] [8].

3.1 Agile QA process

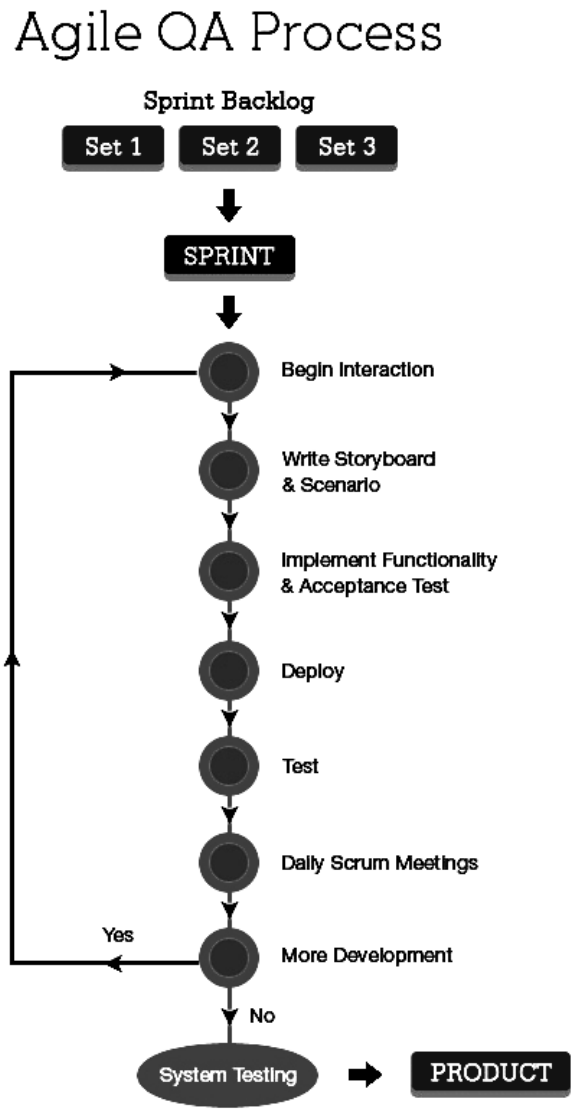

Figure 1: The Agile QA process [7]

In the Agile methods, the work cycle becomes abbreviated. Instead of phases and schedules, it has cadence points which result in completed product increments. Less documentation is required with a focus on code and whether the product piece works. Specification becomes by example with tests for software products based on real examples–a product piece in hand functioning. Every two weeks, the direction of a project receives a review making it easy to steer in other directions as conditions change. Quality becomes part of development and becomes everyone‘s responsibility. By incrementing, inspecting and adapting development cost and time to market becomes reduced. Continual planning of the release optimizes value and maximizes team ability to compete in the marketplace. Included in the process are business people who have a sense of what users need and want. Change becomes welcomed, constant necessary cooperation fosters communication between differing disciplines and adaptability becomes the norm [7] [8].

3.2 Agile Testing Methodologies

Agile Testing has flavors. The method for a particular route of software development must be chosen carefully. Two flavors frequently used by companies are Scrum, and Extreme Programming.

A SCRUM framework emphasizes holistic, flexible strategies where development teams work as units on a common goal. Planning and managing a project is decided by a project’s operation properties and certainties [7]. Product owner is responsible for the delivery and functionality of three core components in each iteration, development team manages and organizes work in a cycle and Scrum Master who removes all obstacles to progress, sets up Sprint meetings, and sets up the teams.

Extreme Programming advocate short development cycles on products that focus on improving the product itself or increasing productivity. It works well in environments where clients have constant change [7] [8].

3.2.1 Phases of SCRUM are as follows

- Test Design: gathers user stories written by the product owner.
- Short Stories: describe functionalities of system under a test and then tests cases are developed based on the user stories with oversight from business analysts and customers.
- Test Execution: involves testing carried out in the lab with tester and developer partnered. Defects are recorded, discussed and tracked in daily meeting. Then retests are set up to fix issues.
- Test Automation: utilizes open source or paid tools to accomplish goals.
- Managing: is the Sprint (cycle of work) meeting at the start of the project that decides scope, reviews logs on product, makes estimates and makes assignments. Daily meetings are 15 minutes where which tasks for the next 24 hours will be done, review the last 24 hours tasks and remove all obstacles [8].

3.2.2 Phases of Extreme Programming are as follows

- Planning: identifies sponsors and stakeholders. It checks infrastructure needs, checks security information and obtains service level agreements.
- Analysis: gathers stories, prioritizes it, and uses the information to make estimates. It then defines the time span needed to accomplish the goal. It makes resource plans for development and QA teams.
- Design: is the breakdown of the tasks needed, how tasks are tested and the regression automation framework.
- Execution: is the work of coding, unit testing, test scenarios, defect reports. It involves the conversion of manual test cases to automation regression test cases. Two reviews exist of mid iteration and end of iteration.
- Wrapping: means granting small releases, regression testing, demos and reviews. New stories are written to address the needs found from the reviews. Process improvements are made based on the end of iteration review comments.
- Closure: launches the pilot program, develops training, and starts production. A SLA guarantee assurance is made with a review of SOA strategy along with production support [8].

3.3 Advantages and disadvantages of switching to agile methods

3.3.1 Advantages

- Testing becomes parallel to the development of an activity so less time taken for testing.
- Iterative and incremental product features may give benefits early in the development.
- Better product through parallel coding, testing and feedback.
- Integrated testing in the lifecycle of the product means inspection and adaptation as a product develops.
- Ability to 'change mind' for the customer. The customer may request a change for the interface, a function, an input or output etc.
- Unpredicted changes can quickly be incorporated into the system.
- Increased visibility of the product owner and product team allows identification of issues early and responding to issues quicker [6] [7] [8].

3.3.2 Disadvantages

- New team members coming on board in the middle of the project or later may find that less amount of documentation is not helpful for them.
- Agile testing and development helps continuous product development. Although this can lead to better products, it can also result in never-ending projects unless improvement goals are prioritized and product delivery deadlines are not set [6] [7] [8].

4 Testing Methods, Levels and Types

In this section the paper will discuss about different testing methods, levels and types which are applicable throughout the software testing lifecycle.

4.1 Methods

There are several approaches / techniques of Software Testing. Although there are many methods of software testing currently available, according to our framework and organization this paper will only include the following methods.

4.1.1 Static

Static testing involves verification and it usually asks or checks "Are you building the thing right?" It is primarily syntax checking of the code or manually reviewing the code, requirements documents, design documents etc. to find errors [1] [9].

4.1.2 Dynamic

Dynamic testing involves validation and it usually asks or checks "Are you building the right thing?" The software should be compiled and executed and input values are given and output values are checked with the expected output [1] [9].

4.1.3 Black Box

Black-box testing treats the software as a "black box" with inputs and outputs [1]. The tester is only aware of what the software is supposed to do, not how it does it. Therefore it is also known as Specification-based testing technique or input/output driven testing techniques.

4.1.4 White Box

In white-box testing or 'glass-box' testing, the internal structures or workings of a program are tested. It is also known as Structure-based technique because here the testers require knowledge of how the software is implemented. In contrast to black-box testing, the tester is concentrating on how the software does it in white-box testing [1].

4.1.5 Visual (GUI)

GUI testing is the process of testing a product's graphical user interface to ensure it meets its written specifications. This type of testing is common when designing web pages where sizes and alignments of images and buttons are checked [10].

4.2 Levels

Each phase of SDLC goes through the testing hence there are various levels of testing. Although there are many levels of software testing currently available, according to our framework and organization this paper will only include the following levels.

4.2.1 Unit Testing

Unit testing is a method by which individual units of source code are tested together with associated control data. A unit is the smallest testable part of an application. For example functions/procedures, interfaces and classes are considered as units. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended [11].

4.2.2 Component Testing

In component testing individual software modules are tested in order to find defects and to verify their proper functionality. Component testing is also known as module and program testing. Component testing may be done in isolation from rest of the system depending on development life cycle model chosen for that particular application [12].

4.2.3 Integration Testing

In integration testing individual software modules are combined and tested as a group to verify they work together without errors. Integration testing is done after unit testing and prior to validation testing. It is done by a specific integration tester or a test team [12].

4.2.4 System Testing

System testing is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and no knowledge of the inner design of the code or logic is required. System testing investigates both functional and non-functional requirements of the software [13].

4.2.5 Acceptance Testing

Once all or most of the defects have been corrected after the system test, the system will be delivered to the user or customer for acceptance testing. It is conducted to determine if the specified requirements are met prior to its delivery. Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well [11].

4.2.6 Alpha Testing

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often used for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing. It takes place at the developer’s site [14].

4.2.7 Beta Testing

Beta testing or field testing, and takes place at customer’s site. The software is installed and tested under real-world working conditions. Versions of the software, known as beta versions, are released to a limited audience - e.g. a sample of targeted customers - outside of the programming team [14]. This is very common in game development.

4.3 Types

A test type is focused on a particular test objective. Although there are many types of software testing currently available, according to the scope and organization of this paper we will discuss only the following types. Testing types are mainly divided into two categories as functional and non-functional testing.

4.3.1 Functional Testing

Functional testing refers to activities that verify a specific action or function of the code. These are usually found in the code requirements documentation, although some development methodologies work from use cases or user stories. Functional tests tend to answer the question of "can the user do this" or "does this particular feature work." Some functional testing types are listed below [1] [11].

- Installation
- Development
- Usability
- Sanity
- Smoke
- Regression
- Destructive
- Recovery
- Automated
- User Acceptance

4.3.2 Non Functional

Non-functional testing listed below are refer to aspects of the software that may not be related to a specific function or user action, such as scalability or other performance, behavior under certain constraints, or security. Testing will determine the breaking point, the point at which extremes of scalability or performance leads to unstable execution. Non-functional requirements tend to be those that reflect the quality of the product, particularly in the context of the suitability perspective of its users [1] [11].

- Compatibility
- Performance
- Security
- Accessibility
- Internationalization / Localization

5 Conclusions

This paper presents an introduction to software testing and its problems. In the future we would like to present two separate papers as solutions to large number of test cases problem using “Combinatorial Testing” and huge testing time problem using “Test Automation” in detail. Combinatorial testing can detect hard-to-find software faults more efficiently than manual test case selection methods. While the most basic form of combinatorial testing - pairwise testing - is well established, and adoption by software testing practitioners continues to increase, the software industry is yet to fully adopt higher strength Combinatorial testing. Test automation paper will demonstrate a hands-on sample industrial web based software project involving functional test automation with SELENIUM tool.

Section two compared manual testing and automated testing. There are both advantages and disadvantages of automated and manual testing. While many say that common sense is supposed to guide you in making the decision of which one to use, common sense won’t lead you unless you have all of the information. Make sure that you consider your time, your resources, and the size of your project as well as the quality of the automated tools you’ll be using and your skills and know-how of your testing team. Once you take these things into mind, you’re more likely to do what’s best for you. Always remember though that combining both is an option. In fact, combining the two may be optimal for canceling out the others’ disadvantages and developing the best software possible.

Agile software development was discussed in section three. Testing discovers information on software errors. This information will be used by the developers becomes to improve the product. Thus the goal of testing products and services is to increase the productivity of a company. Testing must transform into “engineering productivity” approaches. At the heart of all the technology are people with dreams and ideas. Without them, none of the Internet would work. Good practices come about from software testing professionals’ desire to bridge gaps between business, technology, clients and consumers. With more Agile like hybrid methodologies, testing becomes a design criteria; a method to improve productivity and product. Software testing by agile philosophy is a sign that a business and its professionals care to do their work through honest communication with integrity and ethics.

Finally the paper presented testing methods, levels and types - the three main categories of the software testing process. There are several approaches / techniques of software testing and these were discussed under the testing methods. Software goes through testing at each phase of SDLC and hence there are various levels of testing. A test type is focused on a particular test objective and this paper mainly discussed two main types - functional and non-functional - and then listed some testing types under each main type.

References

[1] Myers, Glenford J., Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.

[2] Kung, David, and Hong Zhu. "Software verification and validation." *Wiley Encyclopedia of Computer Science and Engineering* (2008).

[3] Ammann, Paul, and Jeff Offutt. *Introduction to software testing*. Cambridge University Press, 2008.

[4] Qatarun.wordpress.com,. 'Start And Stop Of Testing | Software Testing'. N.p., 2015. Web. 20 Mar. 2015.

[5] Base36,. 'Automated Vs. Manual Testing: The Pros And Cons Of Each'. N.p., 2013. Web. 21 Mar. 2015.

[6] Smartbear.com,. 'What Is Agile Testing? | Smartbear Software'. N.p., 2015. Web. 21 Mar. 2015.

[7] Netsolutionsindia.com,. 'How AGILE Testing Helps In Building Better Software Products'. N.p., 2013. Web. 21 Mar. 2015.

[8] Collins, Eliane, Arilo Dias-Neto, and V. F. de Lucena. "Strategies for agile software testing automation: An industrial experience." *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*. IEEE, 2012.

[9] Zhang, Sai, et al. "Combined static and dynamic automated test generation." *Proceedings of the 2011 International Symposium on Software Testing and Analysis*. ACM, 2011.

[10] Lönnberg, Jan, Ari Korhonen, and Lauri Malmi. "MVT: a system for visual testing of software." *Proceedings of the working conference on Advanced visual interfaces*. ACM, 2004.

[11] Mustafa, Khaled M., Rafa E. Al-Qutaish, and Mohammad I. Muhairat. "Classification of software testing tools based on the software testing methods." *Computer and Electrical Engineering, 2009. ICCEE'09. Second International Conference on*. Vol. 1. IEEE, 2009.

[12] Sawant, Abhijit A., Pranit H. Bari, and P. M. Chawan. "Software Testing Techniques and Strategies." *International Journal of Engineering Research and Applications (IJERA)* 2.3 (2012): 980-986.

[13] Arcuri, Andrea, Muhammad Zohaib Iqbal, and Lionel Briand. "Black-box system testing of real-time embedded systems using random and search-based testing." *Testing Software and Systems*. Springer Berlin Heidelberg, 2010. 95-110.

[14] MacCormack, Alan. "How internet companies build software." *MIT Sloan Management Review* 42.2 (2001): 75-84.

[15] Kuhn, D. Richard, Raghu N. Kacker, and Yu Lei. *Introduction to Combinatorial Testing*. CRC press, 2013.