A Framework to Diminish Software Development Cost by Implementing Smoke and Sanity Testing

By

Ibne Sayad

(172-35-2115)

Supervised by

Mr. Md. Shohel Arman

Senior Lecturer

Department of Software Engineering

Faculty of Science and Information Technology

Daffodil International University

A thesis submitted in partial fulfillment of the requirement for the degree of

Bachelor of Science (B.Sc.) in Software Engineering

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

Fall-2020

APPROVAL

This **Thesis** titled "**A framework to Diminish Software Development Cost by Implementing Smoke and Sanity Testing**", submitted by **Ibne Sayad**, **ID: 172-35-2115** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in software Engineering and approved as to its style and contents.

BOARD OF EXAMINERS

THEIS DECLARATION

The thesis entitled "**A Framework to Diminish Software Development Cost by Implementing Smoke and Sanity Testing**" is done under the supervision of Mr. Md. Shohel Arman, Senior Lecturer, Department of Software Engineering, Daffodil International University.

I declare that this thesis is my original work for the degree of B.Sc. in Software Engineering and that neither the whole work nor any part has been submitted for another degree in this or any other university.

[illegible]

Submitted by

Ibne Sayad

ID:172-35-2115

Department of Software Engineering

Daffodil International University

[illegible]

Certified by

Mr. Md. Shohel Arman

Senior Lecturer

Department of Software Engineering

Daffodil International University

ACKNOWLEDGEMENT

First of all, I’m grateful to the Almighty Allah who created me and gave me the ability to complete this thesis. My parents have played a big role in bringing me this far. They have supported me through tough times and help me to build my career in Software Engineering Sector. I have been able to stand here today with God's undeserved kindness and the blessings of my parents.

I deeply thank my supervisor **Mr. Md. Shohel Arman** for the consistent help of my thesis and research work, through his understanding, inspiration, energy, and knowledge sharing. His direction helped me to finding the solutions of research work and reach to my final theory.

I would like to express my extreme sincere gratitude and appreciation to all of my teachers of Software Engineering department for their kind help, generous advice and support during the study.

My special gratitude goes to my friends, colleagues, seniors, juniors who directly or indirectly have lent their helping hand in this venture.

Ibne Sayad

TABLE OF CONTENTS

APPROVAL i
DECLARATION ii
ACKNOWLEDGEMENT iii
TABLES OF CONTENTS iv
LIST OF TABLES v
LIST OF FIGURES vi
LIST OF NOMENCLATURES vii
ABSTRACT....................viii
KEYWORDS viii

CHAPTER 1: INTRODUCTION....................1-7
1.1 Introduction....................2
1.2 Background....................2
1.3 Software Quality Assurance and Testing....................3
1.3.1 Smoke Testing....................4
1.3.2 Sanity Testing....................4
1.3.3 SDLC Time and Cost....................4
1.3.4 STLC Time and Cost....................5
1.4 Motivation....................6
1.5 Problem Statement....................6
1.6 Research Question....................6
1.7 Research Objective....................6
1.8 Research Scope....................7
1.9 Thesis Organization....................7

CHAPTER 2: LITERATURE REVIEW....................8-9
2.1 Smoke & Sanity Testing & STLC....................9
2.2 Smoke Testing....................9
2.3 Sanity Testing....................9

CHAPTER 3: RESEARCH METHODOLOGY....................10-15
3.1 Proposed Model 11
3.2 Proposed Framework 12
3.2.1 New Build 13
3.2.2 Smoke Testing Phase (STP) 13
3.2.3 Business Requirement Analysis (BRA) 13
3.2.4 Software Requirement Specification Analysis (SRS Analysis) 13
3.2.5 Software Design Specification Analysis (SDS Analysis) 13
3.2.6 Execute Smoke Test (EST) 13
3.2.7 Prepare Test Status (PTS) 14
3.2.8 Prepare Test Report (PTR) 14
3.2.9 Sanity Testing Phase (SaTP) 14
3.2.10 Module Requirement Analysis (MRA) 14
3.2.11 Requirement Traceability Matrix Analysis (RTM Analysis) 14
3.2.12 Sanity Test Plan (STP) 15
3.2.13 Execute Sanity Test (EST) 15
3.2.14 Compare result with SRS, SDS, MRA 15
3.2.15 Set Test Status (STS)....................15

3.2.16 Release For the next Phase……………………………………………………15
3.3 Validate the Framework……………………………………………………….15

CHAPTER 4: EXPERIMENTAL SETUP AND RESULT ……….………16-21
4.1 Experimental Project ………………………………………………………..17
4.2 Data Collection………………………………………………………………...17
4.3 Data Merging…………………………………………………………………..17
4.4 Data visualization……………………………………………………………...17
4.5 Data Explanation…………………………………………………………...17-18
4.6 Module Vs Time Visualization (WUF)……………………………………..…..19
4.7 Module Vs Cost Visualization…………………………………………….…...20
4.8 Describe the Output……………………………………………………………21

CHAPTER 5: CONCLUSIONS AND RECOMMENDATION……………...22-23
5.1 Findings and Contributions………………………………………………….…23
5.2 Limitation……………………………………………………………………...23
5.3 Recommendation for Future works…………………………………………..…23

CHAPTER 6: REFERENCES……………………………………………………..24
6.1 References……………………………………………………………………..25

LIST OF TABLES

Table 1.1: Time & Cost Estimation--- 4-5

Table 1.2: Requirement Traceability Matrix--14

Table 1.3: Time & Cost comparison--18

LIST OF FIGURES

Figure 1.1: Proposed Framework---12

Figure 1.2: Module Vs Time Graph--19

Figure 1.3: Module Vs Coat Graph--20

LIST OF NOMENCLATURE

Terms	Nomenclature
STLC	Software Testing Life Cycle
SDLC	Software Development Life Cycle
SQA	Software Quality Assurance & Testing
WUF	Without Using Framework
UF	Using Framework

ABSTRACT

Today's world needs more digitalization. Software’s are one of the most vital ways to do so. If we want the outcome quicker then we must have to release more software in a short time comparing with the current situation. Statistics show that software testing can grab 15-50% time of total software development time. If we can able to consume some time from the software testing life cycle then the total development time will be reduced. On the other hand, the reduced time will be able to diminish the software development cost. From this idea, we will try to propose a framework based on Smoke and sanity testing methodologies. Smoke testing especially verifies the critical component at the early stage. It's a kind of general health checkup of the whole system. Another one is Sanity testing which is a subset of regression testing. The main objective of the Sanity testing methodology is to check the rationality of the module. So, it's been executed only one particular module at a time. Combining both of these methodologies our proposed framework's main goal will be to reduce the time from STLC. Ultimately, this reduction will play a big role in reducing our software development costs.

Keywords: STLC, SDLC, Smoke Testing, Sanity Testing, Time, Cost

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Software quality assurance and testing are one of the most vital parts of the Software development life cycle. So that, software quality assurance engineers maintain a life cycle in this stage also. Which is called, Software Testing Life Cycle (STLC). Here, there are seven several steps to follow. Those are: - Requirement Analysis, Test Planning, Test Case Development, Test Environment Setup, Test Case Execution, and Test Closure. The Software Quality Assurance engineers maintain these steps vary strictly to gain the best outcome of the software. By following these steps and maintain the quality of work engineers spend a lot of time in the STLC stage.

According to statistics of "jaxenter.com" software testing usually takes 20 percent of the overall development time for a single-component application. It takes 20-30 percent for a two-component application and 30-35 percent time takes for a GUI-based application. But if it's a distributed application with GUI then it can take up to 50 percent of the total software development life cycle time. It’s a huge time to spend on a single stage of SDLC. At the end of the project, it impacts total time and cost.

So, the statistics show that STLC takes a lot of time in the SDLC phase. This huge time spread an impact on SDLC time & cost. It will be so much helpful if we consume some time from the STLC stage. This reduced time will reduce the time of SDLC and by reducing the time the total project cost will be more consumed.

This research paper targets to reduce the time by providing a framework using some software testing methodologies like Smoke & Sanity Testing. With the help of the provided framework, it can surely consume some time on STLC. By consuming the times of the STLC the total SDLC time will be reduced automatically and lesser time will bring lesser cost project.

1.2 Background

The statistics of “jaxenter.com” show that software testing usually takes 15 to 50 % of the total software development time. This huge time slows down the releasing process of the project. On the other hand, the cost gets higher and higher. If some framework can be able to propose by which the time can be reduced on STLC then the project will be released faster and also the total project cost will be more convenient. Based on this fact the idea comes. The proposed framework will be able to reduce the STLC time and the reduced STLC time will reduce the SDLC time and combining both of them the project cost will get lower. There is a famous quote which is "Time is money". If the time gets lower then obviously the project cost will get lower. It will help software companies to develop more projects in a minimum time. With the help of this environment, the world will be digitalized faster.

1.3 Software Quality Assurance and Testing

Control the quality of the product and assure the activities for any kind of business solution or service are very important. If the system being released without any kind of quality assurance and testing then it can bring a huge difficulty to that system. The major goals and attributes to assure the quality of software are: -

Requirements Quality: The correctness and completeness and consistency of the requirements model will have a strong influence on the quality of all products that follow.

Design Quality: Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.

Code Quality: Source code and related work products must conform to local coding standards and exhibit characteristics that will facilitate maintainability.

Quality Control effectiveness: A software team should apply limited resources in a way that has the highest likelihood of achieving a high-quality result.

Software testing is a set of activities that can be planned and conducted systematically. That is why some specific template is being used to follow and maintain the proper procedure. In software testing initially, there different kinds of testing. Like: - Functional Testing, Non-Functional Testing, UI Testing, Security Testing, Database Testing, Performance testing. In Functional Testing, six basic methodologies exist.

Unit testing: The individual unit or the system being tested in this stage.

Smoke Testing: Sometimes it is also called Build Verification Testing (BVT). Here the main critical functional testing is tested.

Sanity Testing: Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes.

Regression testing: Regression testing is a software testing practice that ensures an application still functions as expected after any code changes, updates, or improvements. Regression testing is responsible for the overall stability and functionality of the existing features.

Integration Testing: Integration testing is the phase in software testing in which individual software modules are combined and tested as a group.

Usability Testing: Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end-users.

1.3.1 Smoke Testing

Smoke testing verifies the most important and critical component of the software at the early stage. on the other hand, it confirms that the build version is stable or not. If in this stage the SQA team brings the confirmation to proceed for further testing only then the team moves forward and tests the other module. It's like a general health checkup for the whole system. This testing is also documented while it is testing.

1.3.2 Sanity Testing

First of all, sanity testing is a subset of regression testing. Sanity testing is done to check the new bugs or functionality that have been fixed or not. The main objective of this testing methodology is to check the rationality of the module. So that, it is executed only on a particular module or function. This testing is done by the SQA team but it's not a documented testing. So, it's a kind of specialized health checkup for a particular module.

1.3.3 SDLC Time and Cost

Software Development Life Cycle (SDLC) is a set of processes or steps to develop a software project from start to end. This process is divided into six to eight steps. Usually, this process starts with the planning of the project. Then the business analysts analyze the business requirements and functional requirements. After that, the UI engineers and the software architect design the whole system. When the design has been approved the developers start coding. In this stage, the quality assurance and testing come. The SQA engineers firstly validate the system with the requirements that the system meets the requirements correctly or not. When the validation process has been completed then they start testing by executing some test cases. If the SQA manager declares that the project is ready to release then the DevOps team deploys the project and maintains the project till needed.

Before getting into this project the team prepares an estimation plan. Over there the team analyze the possible modules and set up a time estimation for that project. By calculating the time, the team used to prepare a cost estimation plan. Overall, the lifetime of the project the software company used to follow and maintain that schema.

Items	Working hrs.	Cost/ hrs.	Subtotals	Totals	% Of Totals
1. Project Team	x	x	x	x	59.73 %
Project Manager	1370 h	BDT 625	BDT 8,56,250	BDT 14,51,170	
Team Members	5560 h	BDT 107	BDT 5,94,920		
2. Hardware	x	x	x	x	6.17 %
PC	x	x	BDT 1,20,000	BDT 1,50,000	
Router	x	x	BDT 2,000		
Connectivity & Others	x	x	BDT 10,000		

3. Software	x	x	x	x	3.74 %
Windows 10 Pro	x	x	BDT 52,500	BDT 90,816	
MS Office	x	x	BDT 5,500		
MySQL Server	x	x	BDT 16,500		
Design Tools	x	x	BDT 8,000		
Visual Paradigm Modeler	x	x	BDT 8,316		
4. Testing	280 h	BDT 107	BDT 59,920	BDT 59,920	2.47 %
5. Training & Support	5560 h	BDT 50	BDT 2,78,000	BDT 2,78,000	11.44 %
6. Residence & Food	5560 h	BDT 71	BDT 3,39,760	BDT 3,39,760	13.98 %
7. Others	x	x	x	BDT 60,000	2.47 %
Total Project Cost				BDT 24,29,666	100%

Table 1.1 Time & Cost Estimation

1.3.4 STLC Time and Cost

Life cycle refers to a sequence of changes from one form to another form. Software Testing Life Cycle (STLC) is a process that has some specific steps also. This sequence ensures the quality of the outcome. On the STLC process, every single process or step is carried out in a planned and systematic way. Every step has different kinds of goals.

The STLC starts from the Analysis of the requirements. After analyzing them the SQA team manager prepares a test plan to test the system. Then the team writes test cases based on that plan and modules. When the test cases are ready that time SQA engineers set up the test environment and after that, they execute the test. When the testing gets done that time the engineers announce the test closure.

Here in this cycle, the SQA team estimates the time and prepares a plan. Following the plan, they work until it gets finished. So here the cost analysis also arises with the time.

1.4 Motivation

Nowadays the world is getting more dependent on digitalization. The industry 4.0 revolution is knocking at the door. One the digital world will be a basic need for all people of the whole over the world. In this world, people need more digital solutions. Using those digital solutions, the human life will be easier and faster. So that, people need more digital solutions in a minimum time. On the other hand, the owner of the system needs that system at a minimum cost so that they can easily distribute the solution to the general people. The framework which will be proposed will reduce the time and the cost at the same time. So, the projects will be released faster and because of the shorter time, the project cost will be cheaper. More people will come with a solution and more people will be benefited more than current circumstances.

1.5 Problem Statement

Software quality assurance and testing is that kind of part of SDLC by which the product satisfies the customer. In this stage a lot of time being spent to assure the best quality of the product. The SQA engineers used to follow some guidelines and plans though it takes a lot of time to complete this stage. Many software companies face a huge problem with this kind of situation. Some of the times the companies fail to meet the project deadlines. As a result, the project gets canceled by the owner. It brings a problem for both the software company and the product owner. End of the day the consumer despoiled from the facilities.

1.6 Research Question

Q1. How the framework will affect STLC time and how can it able to reduce the time?

Q2. How the provided framework will reduce the cost?

1.7 Research Objective

The object of this research is very clear and straight. The project time needs to be lesser and the project cost should be cheaper. So that, more people can engage with digitalization and industry 4.0. To achieve that result this paper will use Smoke and Sanity testing methodology. By using those methodologies there will be proposed a framework. Then some data will be included and tested using that framework to verify that the result is bringing the solution or not. Finally, the data will be visualized in the result section.

1.8 Research Scope

Software testing methodologies are used in a lot of software engineering research papers. Those papers use many kinds of methodologies. In this paper, there will be used the main confusing two methodologies of software testing to achieve the desired result. The methodologies are Smoke Testing and Sanity Testing. By using these methodologies there will propose a framework to get the expected outcome.

1.9 Thesis Organization

This paper includes five sections. Those are: Introduction, Literature Review, Research Methodology, Result and discussion, and Conclusion.

- The Introduction section will be discussing the Background, Research objective, Problem Statement, Research Question, and research scope.
- The Literature Review section will discuss the related work to this paper and will try to find the research gaps.
- The Research Methodology section will show the proposed framework for the research and will discuss the research methodology.
- The result and Discussion section will show the result of the methodology with proper visualization and discuss it.
- Finally, On Conclusion section will discuss the final output of the result and future recommendations.

CHAPTER 2
LETARETURE REVIWE

2.1 Smoke & Sanity Testing & STLC

Smoke and sanity testing are some of the most important testing methodologies on STLC. Software companies are using some common testing methodologies but some companies are facing a lot of difficulties to meet the project deadlines. The times are getting overflown and the cost is going higher. Most of the time it's happening because they are forgetting the two essential methodologies which are Smoke and Sanity testing. These are two kinds of methodologies by which the bug identifies at an early stage. It is a fast and cost-effective technique and it's also useful for the development and maintenance of the project [7]. In previous work smoke and sanity testing methodologies used in regression testing also [15]. [14] Sanity testing used in a basic test in ice detection software which was used for Navy ship. [24] Basically after building a version executed and it verifies that the application configuration is correctly done. It's a mandatory process for a performance testing. Smoke test run with one user for every script. If it passes in this test then the actual workload occurs.

2.2 Smoke Testing

[Buchan, W.P] [10] It's been said that the plumbing industry first used the word smoke test in 1875 but some scholars believe that the work comes from electronic hardware testing. As that industry in the software field, smoke testing is used to find the code breaks in the end-to-end testing of the product.

[V.k. Chuhan] [11] Reported that the purpose of smoke testing is to check the latest build is stable or not. If the smoke test passes only then it is handed over to the SQA team for further more testing. Usually, it is done by the SQA team but sometimes it can be done by the developers also. Mostly smoke testing occurs after the new build but if the new test environment is set up also at that time SQA team executes this testing. So that it is also called Build Verification Test (BVT). In Agile methodology, Smoke testing plays an important role. Sometime situation demands that there need to release more than one build in a single day. At that time the smoke testing provides the ensuring to go for further more testing.

2.3 Sanity Testing

[R. Sammi] [7] Reported that the Sanity check or sanity test is a quick and broad level test of systems functionality. One of the most effective matters about the sanity test is If it fails one time then no need to attempt another time. This is another reason for its fastness. A well-managed sanity test approach can able to reduce time and cost. It can also effective for managing the project more effectively.

[Jones] [16] In software schedule and cost estimation sometimes some sanity check rules appear. The resource. scheduling, cost, sizing are the terms that usually come. Those rules are not kind of fully accurate but they can able to give a complete general idea. By implementing those rules of sanity check the final product quality can be improved.

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Proposed Model

Every research paper contains some issues and the researcher proposes a solution for those issues. This paper specifically working on reducing the STLC time and the cost. Here, in this paper, it comes with a framework to diminish the extra time of the STLC. To buildup, this framework has been considered two of the software testing methodologies. Smoke testing and Sanity testing methodology will be used to get the desired result. The research methodology has been separated into a few steps and cycles. The framework will start its work after getting a new build from the developers. While the new build will be approved to test further process, it will be executed in two different cycles. One of the cycles will be executed under Smoke testing and the other one will be executed after Sanity testing. The proposed framework is in figure 3.1. The terms and the methods are being discussed below the framework.

3.2 Proposed Framework

The proposed framework has been visualized in Figure 1.1 below-

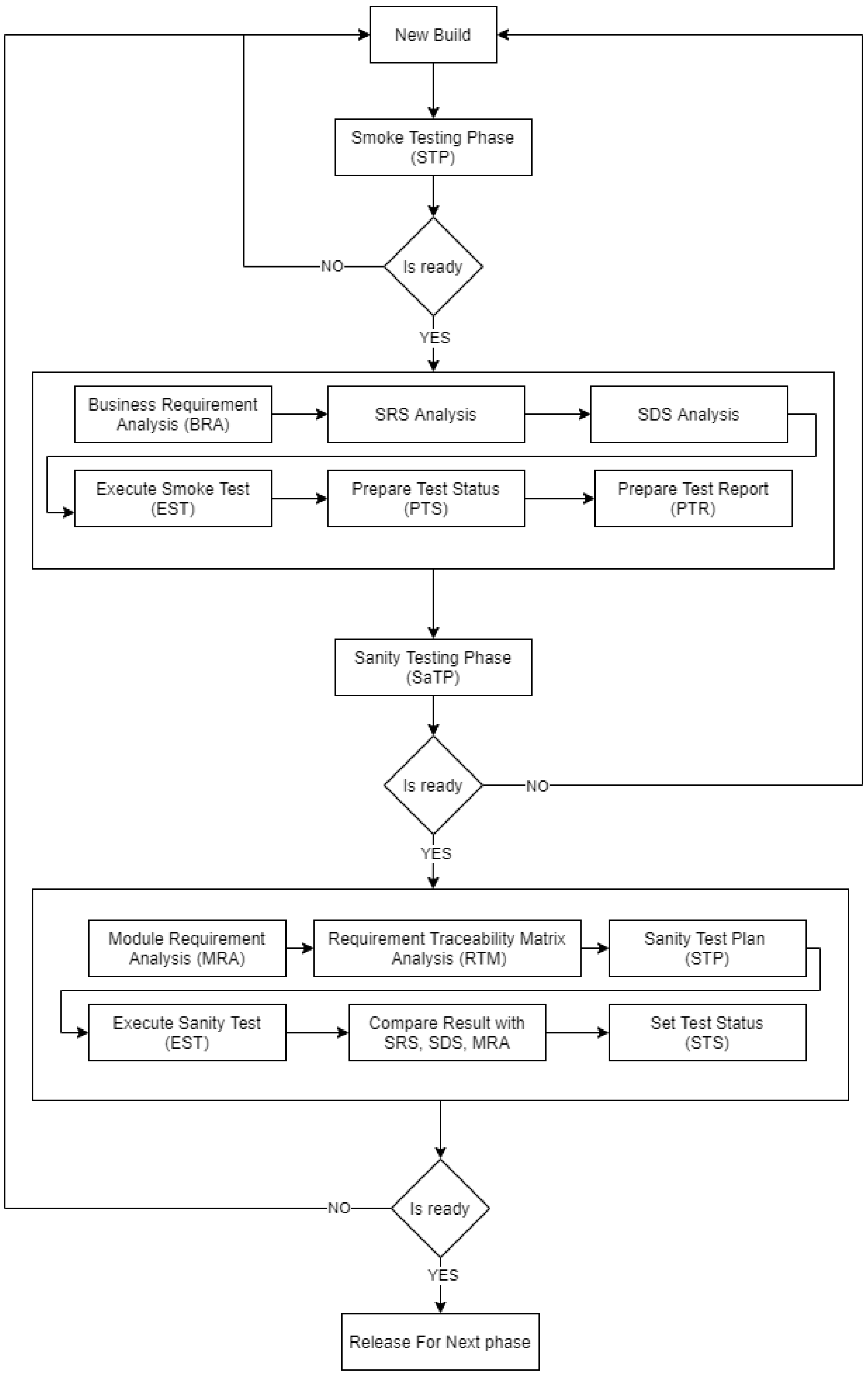
New Build
Smoke Testing Phase (STP)
Is ready
NO
YES
Business Requirement Analysis (BRA)
SRS Analysis
SDS Analysis
Execute Smoke Test (EST)
Prepare Test Status (PTS)
Prepare Test Report (PTR)
Sanity Testing Phase (SaTP)
Is ready
NO
YES
Module Requirement Analysis (MRA)
Requirement Traceability Matrix Analysis (RTM)
Sanity Test Plan (STP)
Execute Sanity Test (EST)
Compare Result with SRS, SDS, MRA
Set Test Status (STS)
Is ready
NO
YES
Release For Next phase

Figure 1.1: Proposed Framework

3 .2.1 New Build

According to Wikipedia "In software development, a build is a process of converting source code files into standalone software artifact(s) that can be run on a computer or the result of doing so." So, build is created to test the collection of code altogether. This build can be released after adding a new module or after fixing bugs.

3.2.2 Smoke Testing Phase (STP)

This is the phase from where the framework will start its work. Here in this phase, it will take a quick checkup about the system that it's ready for the smoke test cycle or not. If this phase has been approved then it will pass it to the cycle otherwise it will return to the build stage.

3.2.3 Business Requirement Analysis (BRA)

Analyze the business requirement plays a vital role in this cycle. Smoke testing's one of the main goals is to check the critical functionality of the system. Without gaining proper knowledge about the system this cycle won't helpful at all. Analyzing the business requirements provides a proper idea about the whole system.

3.2.4 Software Requirement Specification Analysis (SRS Analysis)

Software requirement specification contains a lot of information about the system. Here all the functional requirements of the system stored in a schematical way. Some diagrams like Use case, Activity, Sequence, Class diagrams are presented there. So, in this stage SRS analysis is a must to complete this cycle.

3.2.5 Software Design Specification Analysis (SDS Analysis)

The architectural design of the whole system and the design of the database plays a vital role in Software Design Specification. It's very important to gain complete knowledge about the system design. To do so, SDS analysis is very important for this cycle.

3.2.6 Execute Smoke Test (EST)

After complete the analysis stage now the build is prepared to execute the smoke test. Over here the critical functionality will be checked. There is no need to write the test cases for sanity testing. In a short moment, the smoke testing will be executed and test the build stability.

3.2.7 Prepare Test Status (PTS)

After complete the smoke testing, in this stage, the status of the smoke testing will be analyzed and prepared for the next stage. Here the status and be positive or negative.

3.2.8 Prepare Test Report (PTR)

The test report is important in this stage. The result which has been gotten from the previous stage will be disclosed over this stage. The smoke test result will be analyzed here which will help to decide for the further process either it will go ahead or return to the build.

3.2.9 Sanity Testing Phase (SaTP)

The sanity testing phase will work on a particular module. Usually, sanity testing is also called a subset of regression testing. Over here the module health will be checked quickly. It will be focused on, previously fixed bug on that module that has created a negative impact on this module or not. The other module is hampered or not by this fixing, which will also be checked over this stage.

3.2.10 Module Requirement Analysis (MRA)

Specifically, the requirement of that module that will be tested, will be analyzed properly at this stage. There is no need to analyze the whole system in this phase.

3.2.11 Requirement Traceability Matrix Analysis (RTM Analysis)

The requirement traceability matrix is a table where the requirements and based on those requirements the previously prepared test cases ID and the test status have been shown all together. Reviewing this table helps to gain a complete idea about the previous test result on a particular requirement. A demo of RTM is given below-

Req. No	Req. Description	Testcase ID	Test Status

001	Login to the system	TC-01, TC-02, TC-03	TC-01=Pass
			TC-02=Pass
			TC-03=Pass

002	Create a new course	TC-06, TC-07, TC-08	TC-06= Pass
			TC-07=Fail
			TC-08=No Run

003	Add new students	TC-12, TC-13, TC-14	TC-12=No Run
			TC-13=Pass
			TC-14=Pass

Table 1.2 Requirement Traceability Matrix

3.2.12 Sanity Test Plan (STP)

After complete, the previous stage, here the sanity test plan will be placed over here. The prepared plan will be executed in the next stage.

3.2.13 Execute Sanity Test (EST)

With the help of the previous plan, the sanity test will be executed properly in this stage. After executing the test, the result will be taken to work in the next stage.

3.2.14 Compare result with SRS, SDS, MRA

When the result will be prepared from the previous stage, here that result will be compared with the SRS, SDS, MRA. According to the comparison, the sanity test status will be ready for preparation.

3.2.15 Set Test Status (STS)

The sanity test status will be set in this stage. The status will be gotten from the previous stage. The status can be negative or positive.

3.2.16 Release For the next Phase

After completing the whole framework's process and getting the final test status the framework will finally check either it is ready for the next phase or not. If not then it will pass it to the build. On the other hand, if the status is positive then it will release it for further phase**.**

3.3 Validate the Framework

A research paper needs to validate the proposed model. It's an essential part of that research. To validate our framework, we are considering a software project. On that project, the details of SDLC data and the STLC time and data are properly sorted and analyzed. Over the database, there is presenting a negative set of data that are there without using the framework. On the other hand, there is a positive set of data that are there with using the framework process. The detailed dataset and the compared result will be visualized in the result chapter.

CHAPTER 4
EXPERIMENTAL SETUP AND RESULT

4.1 Experimental Project

For proving the framework, we apply it to a software project. We have considered a total of ten modules there. In each module, we have firstly applied STLC without using the framework. After that, we have calculated the total time it takes. Then another time we have applied STLC on those same modules. Again, we have calculated the total time it takes. Finally, we compared the first scenario data to the second and bring out the result.

4.2 Data Collection

The STLC cycle was monitored properly for each module. The time which was taken by each cycle of the STLC was stored in a database. When the STLC cycle was implemented properly on every module then we had sorted all the data sequentially. After that by using the time, we have calculated the total cost spent for each module and update the database with it.

4.3 Data Merging

When we the desired two attributes of data time and cost then we process those data and sort it properly. So that, we can easily visualize those data in a graph.

4.4 Data visualization

From the sequential dataset, we have represented that huge amount of data in a graph. So that, we can able to overview the total result more shortly. The graph of the result is explained below.

4.5 Data Explanation

The table shows that how the STLC data were synchronized and the time and cost reduced from the cycle. Here in this table, we are considering a single module that was tested in a cycle. The testing life cycle has a total of 6 steps. Each step was usually carried sometimes. Which are represented in the day. The third column of data representing how much of the day it takes to complete this cycle without using the framework. The fourth column of data representing how much of the day it takes to complete this cycle using the framework. After calculating the hourly cost with the third column we got the result of the total Usual Cost.

Usual Cost (WUF) = Usual time (WUF) x Hourly rate

On the other hand, by calculating the hourly cost with the fourth column we got the result of the total cost.

Cost (UF) = Time (UF) x Hourly rate

On the Total row, the fourth column is representing how the time was saved from the previous time and the sixth column is representing how the cost was cased comparing the previous cost.

- Time considered in a day
- UF= Using Framework
- WUF= Without Using Framework

Module Name	STLC	Usual Time (WUF)	Time (UF)	Usual Cost (WUF)	Cost (UF)
M-01 (Deals)	Req. Analysis	2	2	160$	160$
M-01 (Deals)	Test planning	2	2	160$	160$
M-01 (Deals)	Test Case Dev.	4	3	320$	240$
M-01 (Deals)	Test Env. Setup	1	1	80$	80$
M-01 (Deals)	Test Execution	5	3	400$	240$
M-01 (Deals)	Test Closure	1	1	80$	80$
Total:		15	12	1200$	960$

Table 1.3 Time & Cost comparison

4.6 Module Vs Time Visualization

The graph represents that how time impacts the module. In this graph, the first blue line is representing the time comparing with the modules that were not consumed by the framework. The second orange line is representing the time comparing with the modules that were consumed by the framework. After analyzing the graph, we can propose that the framework can able to reduce the STLC time.

Here, without using framework the STLC takes 119 days to complete cycle. By using the framework, it takes 102 days to complete the cycle.

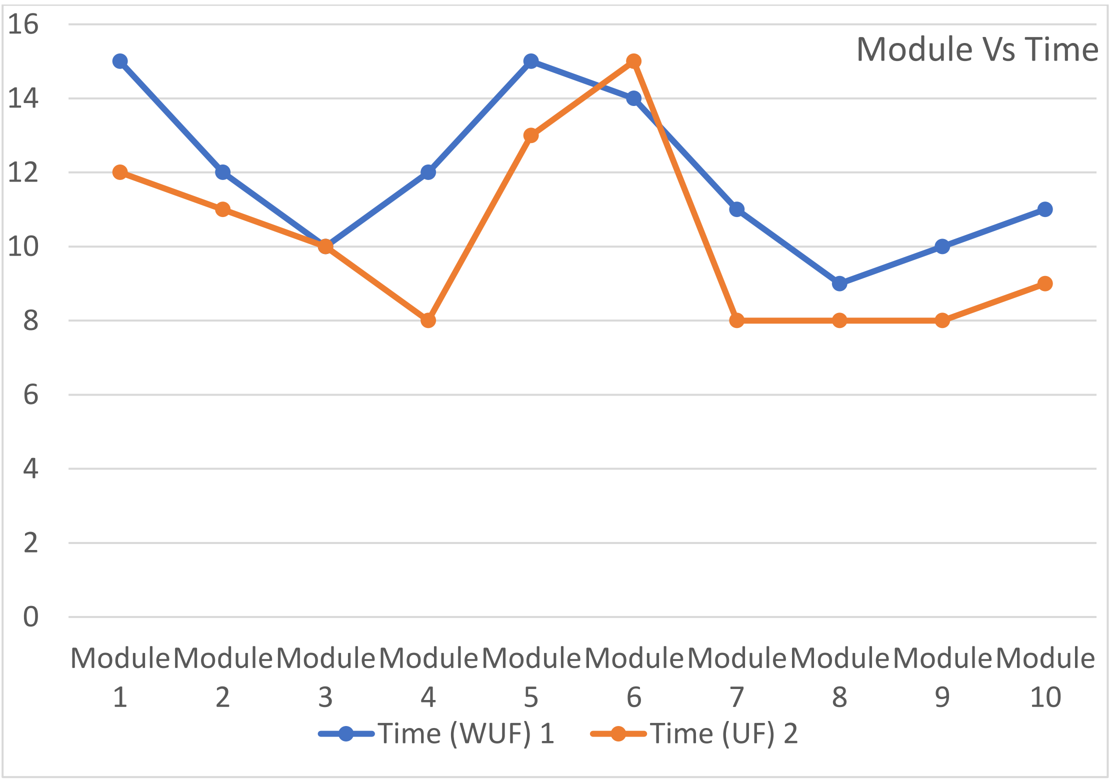

Figure 1.2: Module Vs Time Graph

4.7 Module Vs Cost Visualization

The graph represents that how cost impacts the module. In this graph, the first blue line is representing the cost comparing with the modules that were not consumed by the framework. The second orange line is representing the cost comparing with the modules that were consumed by the framework. After analyzing the graph, we can propose that the framework can able to reduce the STLC cost.

Here, without using framework the STLC takes 11,120$ to complete cycle. By using the framework, it takes 9800$ to complete the cycle.

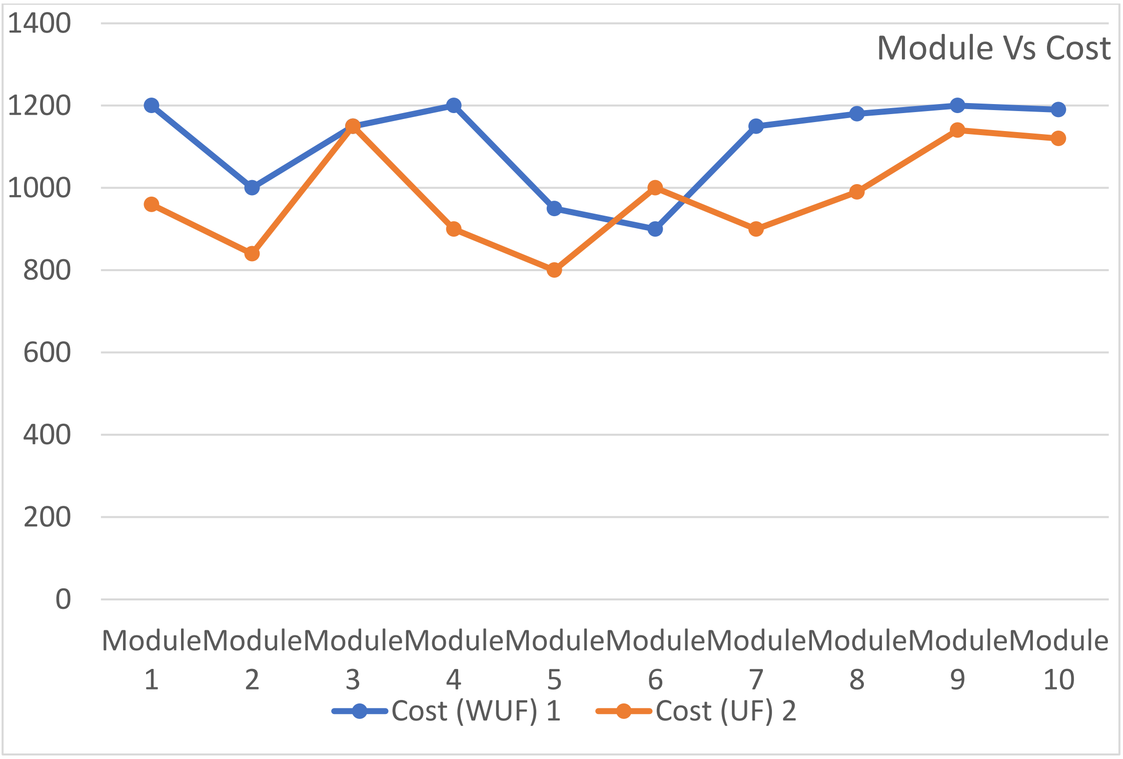

Figure 1.3: Module Vs Coat Graph

4.8 Describe the Output

Considering the two graphs it's clear that, it's possible to consume some time from the Software Testing Life Cycle (STLC) by using the framework. If the time can be consumed from STLC then the Software Development Life Cycle (SDLC) cost will be consumed automatically. Because STLC is a subset of SDLC. How come the time consumed that is represented and discussed in the Topic 4.5 and Table 1.3. The comparison between the module and time was visualized on Figure 1.2.

When the time of the STLC and the SDLC will be reduced then the cost will be reduced automatically. Because the software companies estimate their project cost based on how much of the time it will take. Which was shown in Table 1.1. The data was visualized on Figure 1.3.

So, after considering all of this scenario we can conclude that, the proposed framework on Figure 1.1 can able to reduce the cost of software development. Which is enhancing our research objectives and goals.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATION

5.1 Findings and Contributions

Proposing a completely new framework is always challenging work to do. In this paper, we have tried to achieve our objectives and goals based on two software quality assurance and testing methodologies. Those are Smoke testing and Sanity Testing. Using them we have proposed a framework in Figure 1.1. Based on that framework we have executed it on a software project. The project had a total of ten modules to execute the SQA & Testing. Sometimes the framework had failed to achieve the goal or sometimes it the case was neutral. But almost 80% of the time we were able to achieve our goals. Those are represented in Figures 1.2 & 1.3. So, based on all these factors, we can say that our proposed framework is most often able to achieve the right goal.

5.2 Limitation

In this paper, the research stands on only two SQA methodologies. It's not complete enough to grow a positive impact on the SQA of software engineering as a whole. The work needs to be more versatile and expansion should be more spread on software engineering and software quality assurance and testing. Which will be only possible if the research can take more time. The quality of the work and the result will be more effective and the impact will be more adaptable.

5.3 Recommendation for Future works

It's a long process to research and analysis on a field. Hence, software quality assurance & testing is a huge field to work on. It is also a very significant stage to complete the SDLC. Within a short time, in this paper, the research works on two methodologies of STLC. A framework has been proposed based on these two methodologies and by using the framework and with the help of the dataset the research tried to achieve the desired result. In the future, we will include more SQA methodologies and apply them to a huge level of the dataset. If we got the expected result then will try to propose a complete methodology on software quality assurance and testing to achieve more quality on SQA and software development as a whole.

CHAPTER 6
REFERECES

6.1 REFERENCES

1. Lahouar, S., Rahoman, S.: Design and implementation of an expert system for data sanity checking. In: Proceedings of Southeastcon 1989, Energy and Information Technologies in the Southeast, vol, April 9-12, 1989, IEEE, Los Alamitos (1989), doi:10.1109/SECON.1989.132347

2. Lev-Yehudi, Y., Perry, A.: Implementing Automatic testing is not so automatic: Lessons learned from an implementation experiment in Magic Software Enterprises (1997)

3. Filho, W.P.P.: Quality Gates in Use-Case Driven Development. In: Proceedings of the 2006 international workshop on Software quality, WoSQ 2006 (2006), ISBN:1-59593- 399-9 150 R. Sammi, I. Masood, and S. Jabeen

4. Lassenius, K.R.C.: Pacing Software Product Development: A Framework and Practical Implementation Guidelines (2006), ISBN 951-22-8382-4

5. Jones, C.: By popular demand: Software estimating rules of thumb. Computer 29(3), 116 (1996), doi:10.1109/MC.1996.4859

6. Sundmark, D., Möller, A., Nolin, M.: Monitoring Software Components- A Novel Software Engineering Approach. In: Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC 2004). IEEE Computer Society, Los Alamitos (2004)

7. Rabia Sammi, Iram Masood, and Shunaila Jabeen: A Framework to Assure the Quality of Sanity Check Process. In J.M. Zain et al. (Eds.): ICSECS 2011, Part III, CCIS 181, pp. 143–150, 2011.

8. Mead, N., Stehney, T.: Security Quality Requirements Engineering (SQUARE) Methodology (2005)

9. Li, S., Xu, J., Deng, L.: Periodic Partial Validation: Cost-effective Source Code Validation Process in Cross-platform Software Development Environment. In: Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2004 (2004)

10. Buchan, W.P., " Institution of the Smoke-Test for Drains”.

11. V. K. Chauhan : Smoke Testing: Article February 2014: International Journal of Scientific and Research Publications, Volume 4, Issue 2, February 2014 1ISSN 2250-3153

12. Piprani, B.: Using ORM-Based Models as a Foundation for a Data Quality Firewall in an Advanced Generation Data Warehouse. Springer, Heidelberg (2006)

13. Mellado, D., Fernández-Medina, E., Piattini, M.: A common criteria based security requirements engineering process for the development of secure information systems. Elsevier, Amsterdam (2006)

14. Coulter, R.E.: Ice Edge Detection and Icewater Classification Utilizing the ERS-1 and TOPEX Altimeters. IEEE, Los Alamitos (1994)

15. Onoma, A.K., Tsai, W.K., Poonawala, M.H., Suganuma, H.: Regression Testing in an Industrial Environment. ACM Press, New York (1998)

16. Jones, C.: By popular demand: Software estimating rules of thumb. Computer 29(3), 116 (1996), doi:10.1109/MC.1996.4859

17. Fecko, M.A., Lott, C.M.: Lessons learned from automating tests for an operations support system, Pract. Exper. 00, 1–23 (2002)

18. Zhao, N.Y., Shum, M.W.: Technical Solution to Automate Smoke Test Using Rational Functional Tester and Virtualization Technology. 30th Annual International Conference on Computer Software and Applications Conference (COMPSAC 2006) 2, 367 (2006), doi:10.1109/COMPSAC.2006.166

19. Kaner, Bach, and Pettichord, "Lessons Learned in Software Testing". Wiley Computer Publishing, 2002.

20. Verbauwhede, Ingrid, Patrick Schaumont, and Henry Kuo. "Design and performance testing of a 2.29-GB/s Rijndael processor." Solid-State Circuits, IEEE Journal of 38.3 (2003): 569-572.

21. Ward, C. L., et al. "Design and performance testing of quantitative real time PCR assays for influenza A and B viral load measurement." Journal of Clinical Virology 29.3 (2004): 179-188.

22. Waligora, S. and Coon, R. (1996) Improving the software testing process in NASA’s Software Engineering Laboratory: NASA’s Software Engineering Laboratory.

23. Talby, D., Keren, A., Hazzan, O., and Dubinsky, Y. (2006) Agile software testing in a large-scale project. IEEE Software, 23, 4, 30-37.

24. Rijwan Khan, Mohd Amjad: Smoke Testing of Web Application Based on ALM Tool. ISBN:978-1-5090-1666-2/16/$31.00 ©2016 IEEE 862:

25. Guru99, “Smoke and Sanity Testing - Introduction and Differences”, http://www.guru99.com/smoke-sanity-testing.html