An Integrated Software Test Process Framework: The Case of Selected Ethiopian Software Companies

A Thesis Presented

by

Shimelis Tamiru Duressa

to

The Faculty of Informatics

of

St.Mary's University

In Partial Fulfillment of the Requirements

for the Degree of Master's Science

in

Computer Science

July, 2017

Acceptance

An Integrated Software Test Process Framework: The Case of Selected Ethiopian Software Companies

By

Shimelis Tamiru Duressa

Accepted by the Faculty of Informatics, St.Mary's University, in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science

Thesis Examination Committee:

Asso.Prof.Taye Girma

Internal Examiner

Dr.Getachew Hailemariam (Ph.D)

External Examiner

Mr.Asrat Mulatu

Dean, Faculty of Informatics

July, 2017

DECLARATION

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been duly acknowledged.

Shimelis Tamiru Duressa

____

Addis Ababa

Ethiopia

This thesis has been submitted for examination with my approval as an advisor

Dr.Getahun Semeon Woldemariam (PhD)

____

Addis Ababa

Ethiopia

Acknowledgment

First and for most I would like to thank the Almighty GOD for his unending helps and blessings. Next I would like to appreciate my advisor Dr. Getahun Semeon for shaping up the study and guiding my thoughts and steps to go on the right track. I admire his diligence, timely response and encouragement. I would like to extend my appreciation to my family members and my friend Kirubel Getachew for their moral support during my study. I am also very thankful to all staff members of Informatics faculty St.Mary's University, for their contribution in one way or another for the success of my study. My thanks also goes to all the companies I visited during my research study. Finally I would like to thank St.Mary's University staff members who encourage me throughout my study and the University management at large.

List of Acronyms

SDLC:	Software development Life Cycle
CMMi:	Capability Maturity Model Integration
TMMi:	Test Maturity Model Integration
ISO:	International Standard Organization
IEEE	Institute of Electrical and Electronic Engineering
IEC:	International Electro technical Commission
SQL:	Standard Query Language
ISTQB:	International Standard for software testing and quality Assurance
TMM:	Test Maturity Model
QA:	Quality Assurance
HR:	Human Resource
STMT:	Software Test Methods and Tools
STIF:	Software Test Improvement Framework

LIST OF FIGURES

LIST OF TABLES

Abstract

Software testing is an important phase for quality software development. It constitutes up to 50 percent of the software development time and cost. Although huge amount is invested on software development and software testing is mandatory, most software testing lack formalized processes with the real power and flexibility necessary to adequately test software systems. This is one of the outstanding issues that need to be further explored. In addition, by considering the criticality of software testing and the significant time, effort and cost required, an integrated framework needs to be developed to determine what constitutes an ideal and effective test process in order to ensure quality software product given a resource constrained environment. Although prior studies identified different challenges and proposed solutions, none of them have dealt with developing software testing framework that can guide software testers towards conducting effective and efficient software testing process in a resource constrained environment. By applying mixed method research approach the study assessed the existing practices, processes and challenges of software testing in the Ethiopian software companies and proposed Software Testing Improvement Framework (STIF) which is applicable in a resource constrained environment. The framework is structured in three major areas of challenge having four sub-categories with proposed activities divided in three phases. The framework introduced phased approach of software testing practices involving non-costly activities at the initial phase, implementation of testing with minimalist approach at the second phase and full-scale implementation at the third and advanced phase. This framework has both practical and theoretical contributions.

Key Word: software testing, integrated software testing framework,

Table of Contents

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

Software has become a motivating force. The impact of software on our society continues to be insightful. Most companies seem to understand businesses need software for their daily operations, from hospitals to universities. There is hardly a business environment that can operate in this day and age without software tools. It is important that these tools are tested properly for the business to continue its daily operations. Customers have come to rely on the functionality and accuracy of software. Software testing is perhaps the most important phase of software development life cycle (Koskinen, 2007).

It is an assurance that the software will meet the requirements and function properly. Because of the complex nature of communication between the technical and business professionals, requirement gathering is never a simple task and often the requirements are not properly documented. Under such circumstance proving that the resulting software is fulfilling the business requirements is difficult. Testing or more precisely user acceptance testing (Koskinen, 2007) is supposed to prove that the software fulfills the requirements of the business for which it is being developed. This is not a reclusive or remote problem but is one that various studies have pointed out and almost all software organizations are familiar with. Many studies point out that the major cost in a software project is software maintenance (Borland, 2006).

One factor that can reduce this cost is proper testing. Software testing is a broad aspect that keeps emerging in different stages of software development with different goals. Based on the ever-growing need and application of software products from day to day life to mission critical systems, software testing is one of the most challenging and inevitable issues for companies, organizations, researchers etc (Boeham, 2006).

Different researcher's authors defined software testing from different perspectives. Software testing is the process of executing a program or application, to verify that it satisfies its requirements, detect errors or bugs and to evaluate the features or attributes as well as capabilities of the software (Agarwal, 2010; Alsultannyand Wohaishi, 2009; Tuteja and Dubey, 2012). The software is tested against the final requirements documented to ensure that no feature was left unaddressed. It is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate(Kaner, 2006).Authors also argue that software testing is more than just error detection; testing software is operating the software under controlled conditions :(1) to verify that it behaves “as specified”; (2) to detect errors, and (3) to validate that what has been specified is what the user actually wanted Testing is done at every stage of the software development, in order to verify and validate the software. Therefore, testing is inherent to every phase of the Software Development Life Cycle (SDLC) which is an enforced disciplined approach. Testing is used to ensure software quality that involves verification, validation and reliability of software products. It is the most widely used means to ensure software quality since software quality assurance occupies most concern in the software industry (Shivkumar et al., 2012).

There are many approaches to software testing, but effective testing of complex product is essentially a process of investigation, not merely a matter of creating and following route procedure. It is often impossible to find all the errors in the program. This fundamental problem in testing thus throws open question as to what would be the strategy that we should adopt for testing. Thus, the selection of right strategy at the right time will make the software testing efficient and effective (Khan, 2010).

Software testing comprises of several phases like it begins with unit-testing phase, a stage to look in to separate individual codes and the implementation phase where the software is actually tested and ends with final acceptance of the system with acceptance testing phase. Acceptance testing is conducted to enable a user/customer to determine whether to accept a software product.

Undoubtedly, software quality and its testing still is an art as the principles of software are quite complex. The complexity in software testing arises due to the software coding and programming difficulty. When it comes to software testing, there are many factors to consider for software quality assurance. Whether the selected testing techniques are correct for the specific software, which testing types/methods needs to be used, what are the software requirements, what technique needs to be used for validation and verification, how to do the code coverage testing, whether to keep documentation of testing processes etc. are all essential aspects needed to consider to approach software testing (Maneela et al., 2012).

Bug free software is an illusion and infinite testing is simply not a viable option. But with scientific software development technology, quality control and systematic testing framework, number of bugs could be minimized (Boeham, 2006) Software testing is an important activity to improve software quality. However, it is well known that it is costly (Yang et al., 2008, and Bertolino, 2008). Thus, there has always been a need to increase the efficiency of testing while, in parallel, making it more effective in tester of finding & removing defects. Measuring critical attributes of software testing process provides software developers with further insight in to software testing process framework (Bertolino, 2008) and thereafter, optimal utilization of their resources for testing purpose.

A major concern in software testing is the cost. It is well known that the testing of software is time-consuming and costly process. Testing is performed under controlled environment. Cost is involved in testing and removing of faults and documentation during testing. A vast majority of software projects go over their budget. It’s important to consider the allotted testing budget when prioritizing the features for a given release. Under this we need to consider the following attributes: testing cost, duration of time, resource requirements, and training needs of testing group (Kapur et al., 2014).Testing is a labor-intensive process; due to the fact that its iterative product ensures complete execution of the test scripts. In order for testers to perform efficient testing, it is essential for them to know about the possible cause of the reasons in order to adjourn testing process and also to examine testing efficiency for any software under development. Defining the key activities is the first step.

To recompense for lack of resources, the test process can be accustomed to provide to the limitations set by the operating environment; in fact, there are studies which conclude that adequate testing can be achieved with low amount of resources, even as low as 15 percent of the requested resources(Boehm, 2006). On the other hand, it is also credible to say that software testing can become expensive and wasteful if it is done without any preceding planning. A comprehensive set of the test cases including all possible scenarios outcomes simply cannot be done when software complexity starts rising (Myers, 2004). Finally, there is room for developing test process, if only to steer the testing practices towards better efficiency and effectiveness (Bertolino, 2007). Observing the software testing from the viewpoint of loss of investment, it is easy to understand why organizations should pay consideration to testing activities.

1.2 Statement of Problem

Poor software testing leads to loss of profits and even loss of life (Harter et al. 2012). U.S. organizations lose $60 billion annually because of poor software quality (Harter et al., 2012). The general business problem is that inadequate software testing processes negatively affect software development organizations' profits (Lalit and Joel, 2015). The specific business problem is that some quality assurance or software tester within small software development organizations lack framework or guide lines for successful software quality processes. Designing effective software testing framework can help to overcome some of software testing problems. Armin Beer and Rudolf, (2013), found that lack of experienced personnel is one of the major reasons for companies to lose benefits from the industry resulting significant negative impact in countries economy.

Javed et al., (2012) argued that most software companies in developing countries don't have proper testers. The researchers emphasized that in small software companies a developer is usually fulfilling the responsibilities of tester which is one of the main reasons of lack of software quality. Developer is mostly ineffective when he is reviewing his own code. It will reduce quality and maintainability. On many occasions, small companies just test the functionality of the software and deliver it to the customer. By doing so they save some cost but mostly, the quality of the software is not up to the standard. According to Javed et al., (2012) 85% of losses in software quality are due to lack of skilled professionals. Software companies in developing nations do not follow standards like CMMI since they lack highly skilled specialists, experts and resources like time and budget which results in compromise in software quality.

Most software developing companies or industries give more attention to software development or coding than software testing. Developing countries are struggling with software quality and cannot maintain reputation in International Market (ibid). Standards or the set of guidelines that helps to achieve best results include CMMI and ISO but it is difficult and costly for small Software Development Organizations to follow. As a result a lot of software crisis and failure happen. Historically, software testing has been performed manually and has been error-prone, time-consuming, and costly. Effective software testing model can overcome the deficiencies of manual testing by designing an integrated software framework **(**Nirmala et al., 2013). To the knowledge of the researchers no study has been conducted on software testing challenges and proposing possible solutions by taking a resource constrained environment into consideration. Our preliminary investigation indicated that many of the software companies do not strictly follow standard software testing processes and do not apply standard methods and tools. Despite the rapid expansion of software companies in Ethiopia the qualities of software products is questionable. The practice of software testing also varies from one software company to the other.

There is no prior study that focuses on software testing practices in African context in general and in an Ethiopia situation. The purpose of this thesis is therefore, to assess the existing software testing practices, identify gaps and propose a framework that guides effective software testing process that ensures quality of the software. The study will identify variations in software testing practices and in the use of methods and tools and recommends an integrated framework to improve the process of software testing by considering the prevailing.

Without adequate testing, however, there is a greater risk that an application will inadequately deliver what was expected by the business users or that the final product will have problems such that users will eventually abandon it out of frustration. In either case, time and money are lost and the credibility and reputation of both the developers and software tester and company at large is damaged. More formal, rigorous testing will go far to reduce the risk that either of these scenarios occurs.

When we measure how and when the constraints affect the software testing process it is possible to identify which stage in the software testing process we need to focus on and be able to make the necessary intervention. Therefore, the framework takes into account different factors/constraints that affect the process of software testing and demonstrates how effective software testing can be conducted given the constraints. The proposed framework is based on the experience of software testing teams from the leading Software Development Companies and different standards like CMMi, TMMi and others.

In the course of our research different procedures and standards followed by the leading software companies in Ethiopia will be identified and analyzed which will serve as a basis for developing the framework. Following the framework could enable software companies to achieve effective software testing that can ensure software quality. The framework also contributes to the standardization of software testing processes.

1.3 Research Questions

The study intended to address the following research questions:

1. What are the existing practices and processes of software testing in Ethiopian Software Companies?
2. What are the major challenges that software companies are facing in software testing?
3. What framework can support software companies to perform effective software testing and ensure software quality?

1.4 Objectives

1.4.1 General Objective

The general objective of the study is proposing an integrated framework for software testing that enables software companies to perform effective software testing under a resource constrained context.

1.4.2 Specific Objective

The specific objectives set to achieve the general objective are to:

- assess the existing practices and processes of software testing in selected Software companies in Ethiopia
- identify variations in software testing practices and processes and the major causes for the variations
- identify the major challenges in the existing software testing practices and processes
- propose an integrated framework that address the existing challenges and standardize the process of software testing
- validate the proposed framework

1.5 Research Methodology

1.5.1 Research Approach

In order to meet the general and specific objective the research applies a mixed method approach combining both quantitative and qualitative research methods (Creswell et al., 2011). The combination of different research approaches and datasets is a form of methodological pluralism (Hirschheim, 1985), which itself is a way to increase the validity of the study results (Onwuegbuzie and Leech, 2007). Methodological pluralism allows the researcher to observe the phenomenon from different viewpoints, and also to combine the advantages of two different approaches (Tan and Hall, 2007).

1.5.2 Study population and Sampling

1.5.3 Study population

The overall population of the study is comprised of 10 software developing companies found in Ethiopia located in Addis Ababa, having different capacities of Large, Medium and Small. Researchers assumed that there is difference in the characteristics of the overall selected companies profile in terms of technology usage, staff composition, resource, service coverage and company service year. The selection of the company is based on purposive sampling this is mainly due to the fact that in order to conduct the research company willing is an important factor hence the selected companies have show interest and permission.

1.5.4 Data Collection

The data is collected from primary sources; specifically through questionnaire and observation and interview as an instrument. A questionnaire is prepared based on the stated research questions.

1.5.5 Data Analysis

The collected data from the questionnaire and interview is summarized using statistical methods such as percentage and id represented using tabulation, charts and frequency distribution. The findings from qualitative data gathered through interview will be coded and summarized to triangulate the findings of the survey data.

1.5.6 Validation

Validation on the final output of the research is made though designing questionnaire at some selected software companies who have a better experience and experts to provide us valuable comments

1.6 Scope of the study

Considering the objective of the study this thesis aimed at enhance software testing by carry our survey on some selected software companies who are actively involved in software engineering by identifying their current challenges ,opportunities , practices and propose a frame that can guide them to perform testing activities effectively and efficient in a resource constrained environment.

1.7 Significance of the Study

This study is therefore, aimed at exploring the existing practices and associated challenges in software testing within the context of Ethiopian software companies with the purpose of suggesting an integrated software test framework for ensuring software quality.

The study has also of great significance to Ethiopian software industry. It will benefit also software service provider and software product user hence; this research work will serve as an icebreaker for the software industry in terms of maintaining software quality testing.

1.8 Limitation of the Study

All research projects have shortcomings, threats to their validity and limitation on the scope of their results, and this work is no exception. According to (Nenty, 2009), defined research limitations as the factors that confine and constrain a researcher's study. Simon (2011), indicated limitations could be a source of possible included divulging research limitations. This study contained three limitations. The first limitation was that the accuracy of the results was reflective of the information shared by the participating organizations. The next was the case study sample size was small and may not be representative of the population. The final limitation was the participants limited their experiences to within the participating organization.

CHAPTER TWO

LITERATURE REVIEW

This chapter primarily focuses on reviewing literature which is relevant to the subject of the study. The literature review process aimed at providing a conceptual foundation for the study and gaining a thorough understanding on software testing processes, existing practices, methods, tools, frameworks and challenges in the areas of software testing.

2.1 What is Software Testing?

Abhijit et al., (2012) argued that there are many definitions for software testing. It can be defined as a process of software engineering in which the software is executed with the intent of finding errors, identifying whether it satisfies the requirements and evaluating the features of software Agarwal (2010). Software testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs (Alsultanny and Wohaishi, 2009).Most of the definitions describe testing as a process rather than an activity. This means that instead of a single task, testing is a series of interrelated or interacting activities that lead to a particular result (ISO/IEEE/IEC, 2013).

The following are different interpretations given for software testing; some of them are associated with Beizer’s testing levels (Beizer, 1990).

- Testing is comparing actual and expected behavior: Without that comparison it would be impossible to detect functional failures.
- Testing is detecting failures: Similar to the previous statement, a failure is the observable deviation of the actual from the expected system behavior.

- Testing is managing risks: For many systems, testing cannot be complete and there are only heuristic means of quality measurement. Moreover (Howden, 2006), shows that finding all failures of a system are un-decidable. Thus, deciding when to stop testing is managing the risk of remaining faults. The test effort depends on the kind of remaining possible faults and the corresponding failures. Thus, the test effort necessary for entertainment systems is probably considerably lower than the test effort for critical systems like airplanes or nuclear power plants.
- Testing increases the confidence of testers: Since testing cannot prove the absence of faults, the goal is to remove at least all detected failures.

The definitions often emphasize that the goal of testing is to provide information about the quality of the test item (ISO/IEEE/IEC, 2013) in terms of both functional and non-functional requirements. According to (Juristo et al., 2004), software testing can be considered the most expensive activity in the software development process, and thus, it needs an efficient planning to avoid loss of resources and behind schedule. Software quality on the other hand can be defined as "the degree to which software possesses a desired combination of attributes" (IEEE, 1998). It is a generally accepted fact that a human being can make an error, which then results in a defect in the product. A defect may be behavior in an unintended way or produce an incorrect or unexpected result (Muller &Freedenberg, 2011). According to (ISO/IEEE/IEC, 2013), the purpose of testing is to (1) provide information about the quality of the software, its risk of failure during use, and (2) find defects before it is released to the market. However, one of the benefits of testing is also increasing understanding of the system, especially when developing a very complex system. Therefore, the purpose of testing can be summarized in the following five objectives:

1. Finding defects to help improve the level of quality.
2. Reducing the risk of failures occurring during operation and gain confidence about the level of quality.
3. Improving management decisions by providing information for decision making.

4. Preventing defects by identifying the processes in the organization that need improvements.

5. Gaining insight into the system behavior.

2.2 Importance of Software Testing

Software testing is an essential part of software engineering. The objective of testing is ensuring the quality of the software (ISO/IEEE/IEC, 2013). The information is usually used in decision making and in improving the product quality. However, as the tested systems grow in size and complexity, the same happens to testing and the pool of data it provides. Research has revealed that testing is a major consideration in the software development and maintenance effort and, for many organizations, more time is devoted to testing than any other phase of software development. The use of well-defined testing techniques and methods will minimize the testing effort while maximizing the return on investment.

Testing is an important part of the development process for all projects. It is important to ensure that the quality of software is high and satisfies the customer’s need (El-Halees, 2014).

2.3 Testing Techniques

2.3.1 White Box Testing

White box testing is conducted based on an analysis of internal working and structure of a piece of software. White box testing is the process of giving the input to the system and checking how the system processes that input to generate the required output. It is necessary for a tester to have the full knowledge of the source code. White box testing is applicable at integration, unit and system levels of the software testing process. In white box testing one can be sure that all parts through the test objects are properly executed (Khan, 2010).

2.3.2 Black Box Testing

Black box testing is based on the analysis of the specifications of a piece of software without reference to its internal working. The goal is to test how well the component conforms to the published requirement for the component. Black box testing have little or no regard to the internal logical structure of the system, it only examines the fundamental aspect of the system. It makes sure that input is properly accepted and output is correctly produced. In black box testing, the integrity of external information is maintained. The black box testing methods in which user involvement is not required are functional testing, stress testing, load testing, ad-hoc testing, exploratory testing, usability testing, smoke testing, recovery testing and volume testing, and the black box testing techniques where user involvement is required are user acceptance testing,(Khan, 2010)

2.3.3 Grey Box Testing

Grey box testing techniques combined the testing methodology of white box and black box. Grey box testing technique is used for testing a piece of software against its specifications but using some knowledge of its internal working as well. Grey box testing may also include reverse engineering to determine, for instance, boundary values or error messages. Grey box testing is a process which involves testing software while already having some knowledge of its underline code or logic. The understanding of internals of the program in grey box testing is more than black box testing, but less than clear box testing. (Khan, 2010)

2.4 Software Testing Strategies

2.4. 1.Unit Testing

This testing is performed by respective developers on the individual units of source code assigned areas. The goal of unit testing is to isolate each part f the program and show that individual parts are correct according to requirements and functionality (Grover, 2016). Unit testing is fundamental to the way that people develop software. It refers to testing of separate system's units. Unit testing is usually performed by developers and can be easily automated, providing the base for further application regression testing, checking whether applying small changes and errors correction does not violate system stability. This is how unit testing during development phase is connected with a regression testing, which is performed at maintenance phase after applying changes with new version released (Sen, 2010).

Typical way of managing test for single module is provided in the Figure:

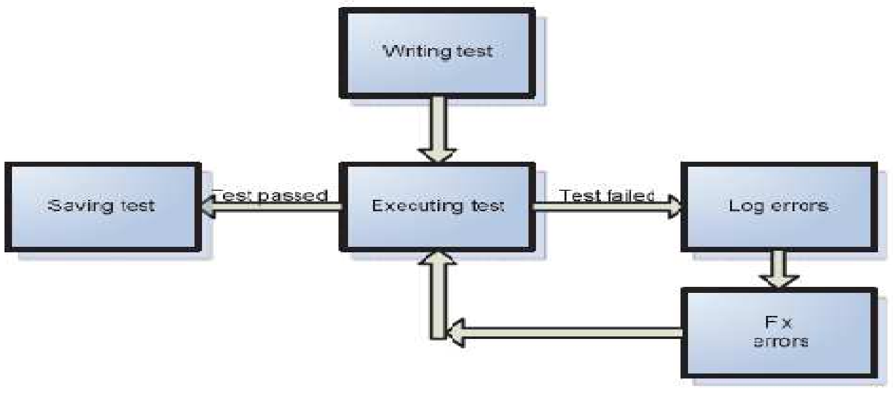

Figure1.1: Managing single test

First developer has to write test and then execute it. If test is passed, developer should save it and continue testing with new test, if test is failed, errors should be logged, then fixed and test should be executed again till the moment it will be passed. Logging errors and saving old tests are very valuable actions, because they will help in fixing errors in future.

2.4.2 Integration Testing

It is combined part of an application to test whether all functions are correct or not. This technique is based on: Top down integration and bottom up integration. Top down means the testing begins with unit testing. Bottom up means highest modules are tested first (Grover, 2016).

The purpose of integration test is to make sure that interaction in a system between multiple components and data transfer work properly without flaws. There is more than just one level where integration testing takes place. The developers usually are responsible for the integration testing at the lowest level; they are the ones who check if all units work well together. The developers can also do the higher level integration tests, but usually software testers are the ones who take the lead at this stage, since the biggest focus is on the interfaces which are usually software testers' specialty; to look if all the units of the system work properly together. As the integration test is about many different parts of a system working together, which is done usually near beta testing when it's possible to combine the newly created software with e.g. old system parts which are meant to interact with the new software. In some cases these can be done with a customer, since they have more knowledge of how the new system should work. Integration tests require quite a lot from the testing team, as it may continue very long, even after the deployment of the software, and because of that, the testing team has to try to check every aspect of the system in order to have the upper hand over the customer(Loveland et al.,2005).

But it must be taken into a notice, that usually even the most advanced or qualified test team hardly ever succeed to create a test environment which would be identical to the customer's own system. It can be similar from the outside, but the units can differ. This is the point where good relationships are required with the customer, and a mutual trust must be achieved in order to have the customer's knowledge of how the their own system is created, how the units work, and how they're configured. This knowledge helps the test team to create an environment as similar as possible to the customers own environment, helping them to understand how it functions, and make better tests(Loveland et al., 2005), the structure of which is provided on figure: the bottom up and top down integration testing

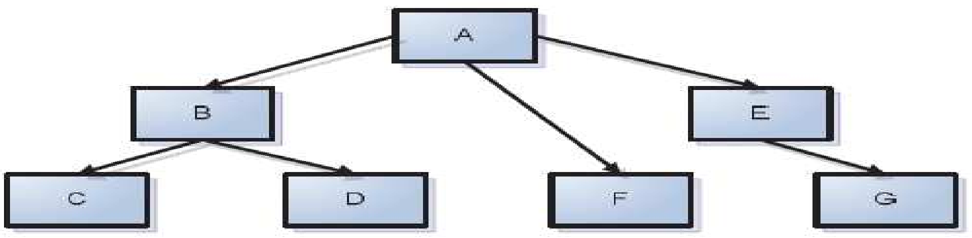

Figure 2.1 provides the basic structure of the program – it shows main modules of the program and how they are related. Arrow going from module A to module B means that module B is used in module A. The lowest module at this example are C,D,F,G, the highest – A. Bottom-up testing can be started from any of the modules at the lowest level, i.e. from C, top-down testing starts from the highest module A.

2.4.3 System Testing

Where integration testing focused more on how different units in the system work together, system testing has a lot vaster area of tests involved. In a medium sized software company there are hardly ever several test teams, so at this point they have a lot on their hands. System test includes usually functional tests, load tests, performance tests and reliability tests, not to mention that system testing can last very long, depending on resources and needs. So these are probably the reasons why software testing is often generalized as system testing. And since the system test is considerably large part of the whole testing of the software, most test cases and sets are stored and reused when regression testing takes its turn(Loveland et al.,2005).

For system testing there has to be a system test plan. This plan is usually created by the manager of the testing team, or a senior software tester who is the one responsible on coordinating users' and developers' focus on areas that matter. Sometimes the test team collaborates with the development team and the customers to have a shared view of the plan. There are some prerequisites for the system test plan: the requirements from the customer which have been modified into specifications of the system, and then the software design documentation. The writing of the system test plan can be started earlier, but it is not recommended (Loveland et al., 2005).

No developer or tester is perfect, and that is a sad fact. After long periods of tests, there comes a time when the customer gets his hands on the software and finds a defect. This can happen during a life cycle test in approval tests with the customer, or later when the deployment has already been made. Once a defect is found, it is reported and studied until it is identified and provided with a fix. The same kind of tests similar to the system tests have to be performed when the fix is applied, and a new release of the software has been published. And these things are not to make haste with, since now that the customer has seen a flaw in the system, they will only get more accurate and agitated with the test, and testers (Loveland et al.,2005).

2.4.4 Acceptance testing

Acceptance testing (also known as user acceptance testing) is a type of testing carried out in order to verify if the product is developed as testing is generally carried out by a user/customer where the product is developed externally by another party. Acceptance testing falls under black box testing methodology where the user is not very much interested in internal working/coding of the system, but evaluates the overall functioning of the system and compares it with the requirements specified by them. User acceptance testing is considered to be one of the most important testing by user before the system is finally delivered or handed over to the end user. Acceptance testing is also known as validation testing, final testing, quality Assurance testing, factory acceptance testing and application testing etc. And in software engineering, acceptance testing may be carried out at two different levels; one at the system provider level and another at the end user level per the standards and specified criteria and meet all the requirements specified by customer. This type of testing reduces testing costs by supporting the test process with a range of software tools (Isha and Sunita, 2014).

2.4.5 Regression testing.

Regression testing is applied to code immediately after changes are made. The goal is to assure that the changes have not had unintended consequences on the behavior of the test object. Regression testing is applied during development and in the field after the system has been upgraded or maintained in some other way. Good regression tests give confidence that the software can change the object of test while maintaining its intended behavior. Regression testing is an important way of monitoring the effects of change (Stuart, 2011), Software testers normally choose a set of test case from the test suite designed for the product to be automated for the regression testing (Mei et al., 2009)

2.4.6 Security Testing

Designing and testing software systems to ensure that they are safe and secure is a big issue facing software developers and test specialists. Security testing evaluates system characteristics that relate to the availability, integrity, and confidentiality of system data and services. Users/clients should be encouraged to make sure their security needs are clearly known at requirements time, so that security issues can be addressed by designers and testers. Physical, psychological, and economic harm to persons or property can result from security breaches. Following are the main aspects which Security testing should ensure (Arvinder et al., (2007), Confidentiality, Integrity, Authentication, Availability, Authorization, Non-repudiation, Input checking and validation and SQL insertion attacks.

2.5 Automated testing

Automated testing is a form of test automation that uses software to control the execution of tests (Hass, 2008). Test automation involves automating a manual test process already in place by software testers, in which the testers program the automated tests to compare actual test results to expected test results along with setting up test conditions, controls, and reporting functions (Hass, 2008). The goal in automating the test is to reduce time for manual execution of a test and to allow test cases to be rerun for regression purposes (Hass, 2008).Automated Testing is a testing process where software testing tools conducts pre-scripted tests on software to verify whether all the functionality are working properly, all the requirements for the software application are met properly, the version of the software application is bug free and updated etc.

Why Automated Testing?

With the advent of testing tools, automated testing has become more and more popular. There are a lot of online software testing tools available on the Internet. Companies also use customized testing tools for their software which takes time, effort and money to build. But for quality assurance control, to be able to run tests repeatedly, to use the same type of tests and test results later, and to compare test results automated testing tools are inevitable. One can use one testing tool for all test types and levels, or different ones for different types and levels. Regardless of the size of company or project, the use of automated testing is increasing because it makes testing easy and helps to deliver almost flawless end product in shorter time period (Leonardo, 2012). A research from 2009 depicts that, large scale company like Microsoft is using NUnit automated unit testing framework for unit testing purpose (Christer, 2012).

2.6 Cost of Software Testing

Software managers have spent an estimated 50% to 80% of development budgets to detect and fix defects (Gupta & Bhatia, 2010, Silva &Someren, 2010). Specifically, managers set aside funding from the overall project budget for assessing the product and resolving the defects that the testers find (Hass, 2008). Whether such an investment can be considered a definite contributor to the project depends on whether the test investment produces a positive return, fits within the overall project schedule, and has quantifiable findings and defect removal (Gupta & Bhatia, 2010, Hass, 2008, Silva & Someren, 2010).

To calculate the cost of product quality, a principle of quality cost is used in a formula:

*Cost of quality = Cost of conformance + Cost of nonconformance (*Phillips et al.,2007). The cost of conformance includes prevention costs and appraisal costs. Prevention costs include money spent on quality assurance such as training, requirements and code reviews, and other activities that promote good software. Appraisal costs include money spent on planning test activities, developing test cases and data, and executing those test cases once. The nonconformance cost includes internal failures and external failure (Ibid). Internal failure costs include expenses that arise when the unit test cases fail the first time they are run by a programmer and will increase after the product undergoes formal testing by the test team.

As a tester researches and reports defects, a programmer confirms and fixes the defects. The release engineer then produces a new software release and the new release must be retested by the tester to confirm the fix as well as to conduct appropriate regression tests to ensure that nothing else is broken by the fix.

The costs of external failure are incurred when customers find defects (Phillips & Pulliam, 2007). The costs to fix defects found by customers are higher than the cost of defects found by programmers and testers because not only are the same costs described for tester-found defects incurred, but technical support, sales, and marketing cost overheads are also incurred. Other intangible costs also exist when there are unsatisfied customers because of the damage that occurs to the company image as a result of the poor quality product (Bertolino, 2007).

To save both internal and external failure costs, defects must be searched and removed early in the development process. The V-model for software development shows that testing process can take as long as or even longer than the development process because test planning often starts as early as the design steps of the project development process (Hass, 2008).

As a result, software testing can cost more than 50% of the development cost (Bertolino, 2007). Because testing costs money, takes a long time to complete, and does not help in the actual building of the product, the process of negotiating for a software testing budget can be difficult for some project managers (Phillips & Pulliam, 2007). Hence, corporate software tests and quality measurements for testing processes continue to be deficient, resulting in a high number of product defects discovered after the release of a product (Glass et al., 2006, Humphrey, 2008), Jones, 2008).

Managers of software development and testing organizations are always interested in a contained, cost-effective testing effort that can ensure the discovery and removal of a sufficient amount of defects (Mockus et al., 2009). As a result, additional testing research is needed to improve quality while minimizing the costs. A cost-effective perspective means testing until the optimum point is reached, which is the point where the cost of testing no longer exceeds the value received from the defects uncovered.

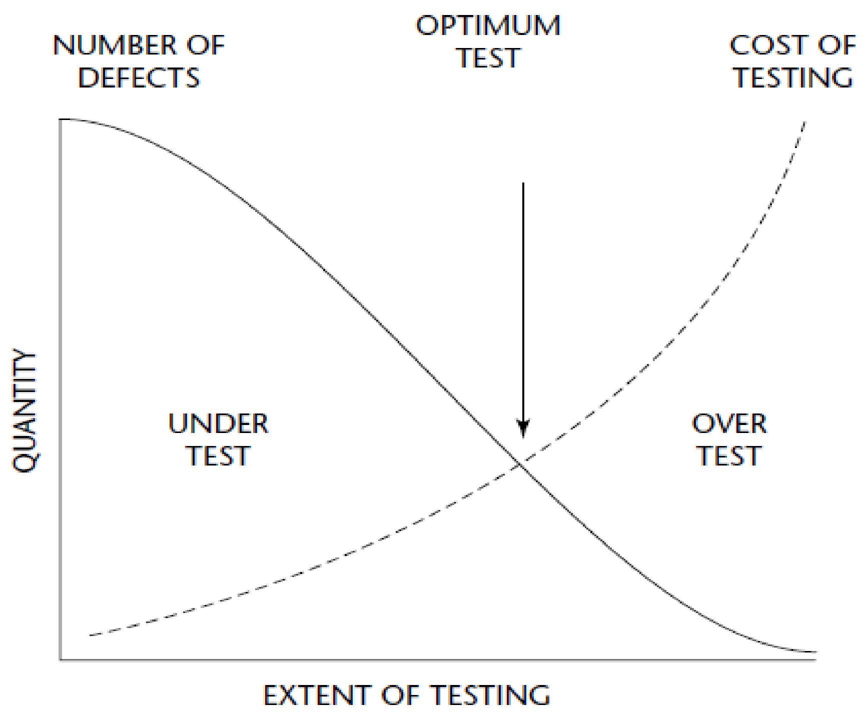

Fig 2.3 Test Cost Curve (Wiley, 2006)

Organizations must try to establish a basis to measure the effectiveness of testing. This makes it difficult for the individual systems analyst/programmer to determine the cost-effectiveness of testing. Without testing standards, the effectiveness of the process cannot be evaluated in sufficient detail to enable the process to be measured and improved. The use of a standardized testing methodology provides the opportunity for a cause and effect relationship to be determined. In other words, the effect of a change in the methodology can be evaluated to determine whether that effect resulted in a smaller or larger number of defects. The establishment of this relationship is an essential step in improving the test process. The cost-effectiveness of a testing process can be determined only when the effect of that process can be measured. When the process can be measured; it can be adjusted to improve the cost-effectiveness of the test process for the organization. (Wiley, 2006)

2.7 Challenges in software testing

Testing of software today is a challenging activity in software system projects. Katherine &Alagarsamy, (2012) defined testing as “one of the five main technical activity areas of the software engineering lifecycle that still poses substantial challenges. It’s generally believed that unidentified bug at the time of software testing is one of the biggest challenges that software testing is facing currently. The following sub-section discusses the major challenges that software companies face in conducting effective software testing.

2.7.1 Inability of Testing to Detect Defects

Some testing activity is unable to detect uncover defect due to few constraints such as lack of experiences of testing team, no automated tool, lack of knowledge and others. According to Ahamed, (2009) he mentioned that unidentified bugs could cause future software failure. The problem of performing a software testing is for defect detection which software can only suggest the presence of flaws, not their absence. The essence of software testing is to explore the whole system in search of bugs or for defect detection, and reliability estimation. Furthermore, Quadri (2010), also supported the statement that testing can be used to show the presence of errors, but never to show their absence. Therefore, inability of software testing to track future error (defect) becomes a big challenge of software system testing.

2.7.2 Lack of Skilled Manpower

Javed et al., (2012) argued that most software companies in developing countries don’t have proper testers. They also emphasized that in small software companies a developer is usually fulfilling the responsibilities of tester which is one of the main reasons of lack of software quality. Developer is mostly ineffective when he is reviewing his own code. It will reduce quality and maintainability. On many occasions, small companies just test the functionality of the software and deliver it to the customer. By doing so they save some cost but mostly, the quality of the software is not up to the standard. According to Javed et al., (2012), 85% of losses in software quality is due to lack of skilled professionals. Software companies in developing nations do not follow standards like CMMI since they lack highly skilled specialists, experts and resources like time and budget which results in compromise in software quality (Ibid.). (Ramos, Esteban & Guzman et al., 2012) suggested that it is very important for software testing team members to develop other essential competences to perform testing activities efficiently in a global context and in order to solve current issues facing software testing.

2.7.3 Lack of testing framework

Another issue is lack of testing framework that can guide new software tester to refer as a guideline (Javed et al., 2012), According to (Richa, 2013) unfortunately, there is no well-defined software test framework that allows organizations to assess step or procedures to perform testing activity in a software project. Oriordain, (2008) identified the importance of software test frame work for the company he suggests that an explicitly defined test framework can facilitate the planning, organization and execution of all test activities within a company". He warns that even an implicitly defined test policy or complete lack test frame work.

2.7.4 Time Duration

Time duration is also an important factor that affects the quality of software. Mostly, software testing team has very tight schedule (Javed et al., 2012).In testing activity, the time allocation is very limited. More than 60% of the time is allocated for development phases while time for testing activity is less than 40%.Any project has to be completed within a given time. Only then the project becomes viable and hence profitable. If there is delay in completing the project due to unforeseen circumstances, it leads to various other complications and sometimes even liquidated damages from the client (Alsultanny and Wohaishi, 2009).

2.7.5 Poor documentation

Poor documentation always creates more challenges to software testing activity and the communities (Mansor, 2012). The example of poor documentation such as lack of information provided, poor instructions, appendices and others in the documents. It is difficult to the users or to the software testing community to refer to the procedures or guidelines if problem with poor documentation is continuously occurs. In addition, a failure to anticipate the reader's obstacles, questions, and environment adds more problems to software testing activity. In general, testing documentations are written for the authors and their environment not for the users. However, documents should be suitable for the users in order to cope with this challenge facing software testing to some extent. It's very important to consider user documentation and system documentation as its necessary during the development of the software to avoid future challenges that might occur during the testing of the software (Aregbesola et al., 2011)

2.7.6 Lack of Planning and Coordination

Poor planning and coordination were identified as one of the reason for project failure (Haugest, 2009). Planning and the coordination plan should be done clearly and accurately before project start. Software testing activity fails due to poor planning before executing testing cases. Planning for testing of software should be considered in prior phases of software project development (Haugset, 2009).When testing is not supported with planning the last stages of the project it leads to failure of project.

2.8 Software Test Process

The preparation actions, actual testing work and test reporting done in a software project formulates a test process. For example, in ISTQB Glossary (ISTQB, 2007) of terms used in software engineering, the software process is defined as follows:

> The fundamental test process comprises test planning and control, test analysis and design, test implementation and execution, evaluating exit criteria and reporting, and test closure activities.

Further, the working draft of the ISO/IEC 29119 standard (ISO/IEC, 2010) specifies three layers of testing process, dividing the process of conducting testing to following components:

(1) Organizational test process**:** Includes test policy and test strategy

(2) Testing management processes: includes test planning, test monitoring and control and test completion.

(3) Fundamental test processes**:** are further divided into static test processes, which constitute universal activities done with all test cases such as test reporting or case design, and dynamic test processes, which constitute changing activities, such as configuring of different tools or executing a test case. Related to these layers are the four different concepts of test process, which are defined in the (ISO/IEC,29119) glossary as follows:

Test policy: high level document describing the principles, approach and major objectives of the organization regarding testing.

Test strategy*:* A high-level description of the test levels to be performed and the testing within levels for an organization or program (one or more projects).

Test management: The planning, estimating, monitoring and control of test activities, typically carried out by a test manager.

Test execution: (1) the process of running a test on the component or system under test, producing actual result(s). (2) Processing of a test case suite by the software under test, producing an outcome. 3) Act of performing one or more test cases (ISO/IEC/IEEE, 2010) Systems and Software Engineering Vocabulary (ISO/IEC/IEEE, 2010)

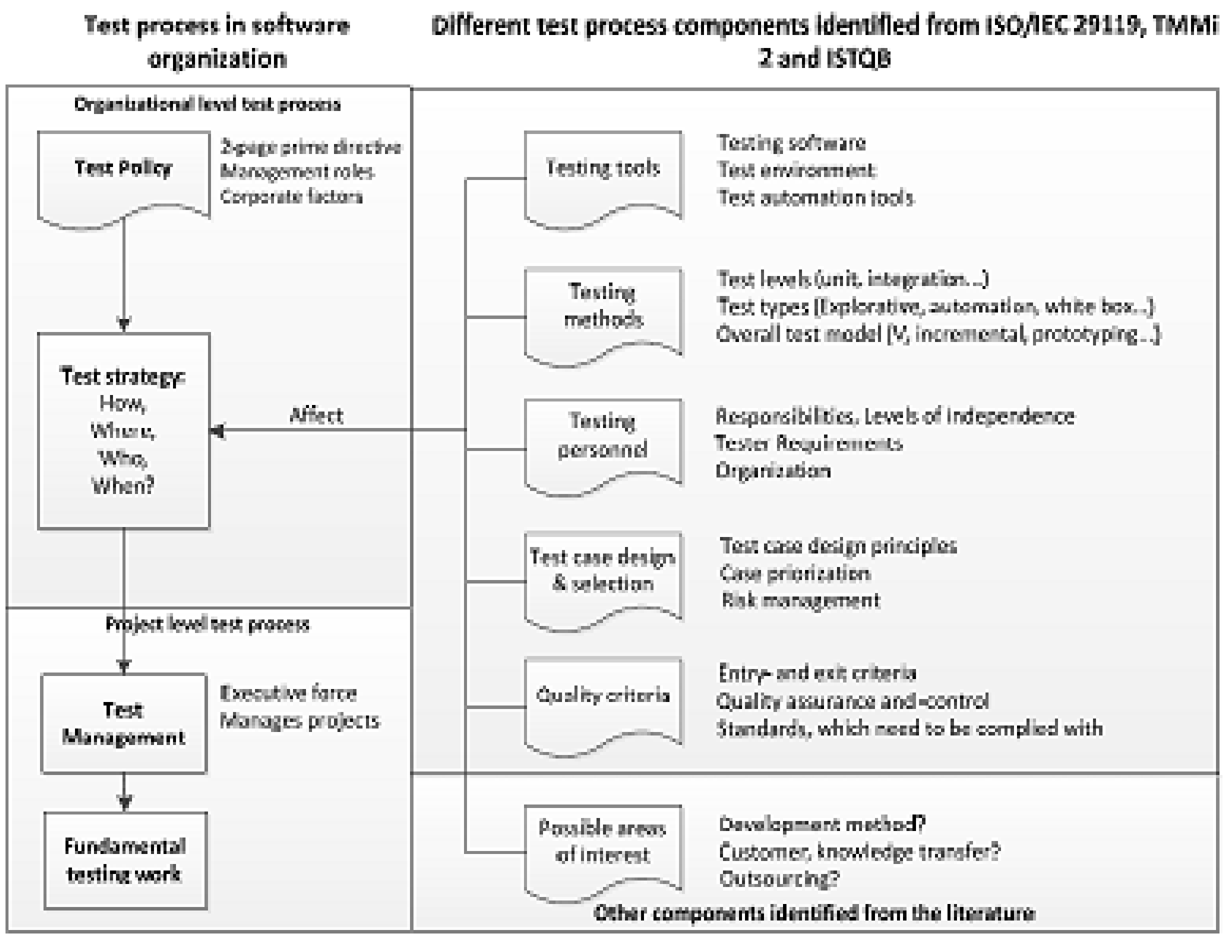
Test process in software organization
Different test process components identified from ISO/IEC 29119, TMMi 2 and ISTQB
Organizational level test process
Test Policy
2-page prime directive
Management roles
Corporate factors
Test strategy: How, Where, Who, When?
Affect
Project level test process
Test Management
Executive force
Manages projects
Fundamental testing work
Testing tools
Testing software
Test environment
Test automation tools
Testing methods
Test levels (unit, integration...)
Test types (Explorative, automation, white box...)
Overall test model (V, incremental, prototyping...)
Testing personnel
Responsibilities, Levels of independence
Tester Requirements
Organization
Test case design & selection
Test case design principles
Case priorization
Risk management
Quality criteria
Entry- and exit criteria
Quality assurance and -control
Standards, which need to be complied with
Possible areas of interest
Development method?
Customer, knowledge transfer?
Outsourcing?
Other components identified from the literature

Figure 2.4: Different test process components and the levels of the ISO/IEC 29119 Model in the test process of the software organization (taken from TMMi foundation 2010)

2.8.1 Test Planning

Test planning is one of the keys to successful software testing. (Burstein, 2009), suggests that test planning is an essential component of a test process as it ensures that the process is repeatable, defined, and managed. (Gelperin, 2008) also supports this view and state that test planning contributes significantly to improve the test process. Companies are highly encouraged to incorporate a test planning in each phase.

The goal of test planning is to take into account the important issues of testing strategy, resource utilization, responsibilities, risks and priorities. Test planning issues are reflected in the overall project planning. The test planning activity marks the transition from one level of software development to the other, estimates the number of test cases and their duration, defines the test completion criteria, identifies areas of risks and allocates resources. Also identification of methodologies, techniques and tools is part of test planning which is dependent on the type of software to be tested, the test budget, the risk assessment, the skill level of available staff and the time available (Tian,2005). The output of the test planning is the test plan document. Test plans are developed for each level of testing. The test plan at each level of testing corresponds to the software product developed at that phase. According to (Tian, 2005), the deliverable of requirements phase is the software requirements specification. The corresponding test plans are the user acceptance and the system/validation test plans. Similarly, the design phase produces the system design document, which acts as an input for creating component and integration test plans.

2.8.2 Test Design

The Test design process is very broad and includes critical activities like determining the test objectives (i.e. broad categories of things to test), selection of test case design techniques, preparing test data, developing test procedures, setting up the test environment and supporting tools. Determination of test objectives is a fundamental activity which leads to the creation of a testing matrix reflecting the fundamental elements that needs to be tested to satisfy an objective. This requires the gathering of reference materials like software requirements specification and design documentation. Then, on the basis of reference materials, a team of experts (e.g. test analyst and business analyst) meet in a brainstorming session to compile a list of test objectives. (Pressman, 2005)

2.8.3 Test Execution

As the name suggests, test execution is the process of running all or selected test cases and observing the results. Regarding system testing, it occurs later in software development lifecycle when code development activities are almost completed. The outputs of test execution are test incident reports, test logs, testing status and test summary reports (Craig, 2002).

2.8.4 Test Review

The purpose of the test review process is to analyze the data collected during testing to provide feedback to the test planning, test design and test execution activities. When a fault is detected as a result of a successful test case, the follow up activities are performed by the developers. These activities involve developing an understanding of the problem by going through the test incident report. The next step is the recreation of the problem so that the steps for producing the failre are re-visited to confirm the existence of a problem (Craig, 2002).

2.9 Software Industry and Software Testing in Developing Countries

Developing countries are struggling with software quality and cannot maintain reputation in International Market (Javed et al., 2012). Standards are the set of guidelines which help to achieve best results. The standards and procedures include CMMI and ISO but it is difficult and costly for small Software Development Organizations to follow the standards (Javed et al., 2012).

For the software industry in developing countries to grow strong and be a viable source of external revenue, software assurance practices have to be taken seriously because its effect is evident in the final product. Moreover, quality frameworks and tools which require minimum time and cost are highly needed in these countries (Sowunmi et al., 2016).

2.10. Test standards ISO /IEEE

What are standards?

According to ISO, standards are “Guideline documentation that reflects agreements on products, practices, or operations by nationally or internationally recognized industrial, professional, trade associations or governmental bodies”. They are guideline documents as they are not compulsory unless mandated by an individual or an organization so although there is a widespread perception that standards are imposed on people and organizations, in fact that depends on how they are used. If specified in a contract then they can define requirements, but this depends on the users. Standards in general have been shown to provide increased productivity and profitability for businesses of all sizes – and, perhaps more surprisingly, enhanced innovation.

Test Standards

Unhappily, up until now there has been no definitive software testing standard. Consumers of software testing services cannot simply look for the ‘badge of compliance’ and testers have no single source of good practice. There are many standards that touch upon software testing, but many of these standards overlap and contain what appear to be contradictory requirements with conflicts in definitions, processes and procedures. Given the current conflicts and gaps, it seems clear that the ideal solution would be to develop an integrated set of international software testing standards that provide far wider coverage of the testing discipline. And ideally this initiative would not re-invent the wheel, but build upon the best of the available standards; thus the motivation for the ISO/IEC/IEEE 29119 set of standards.

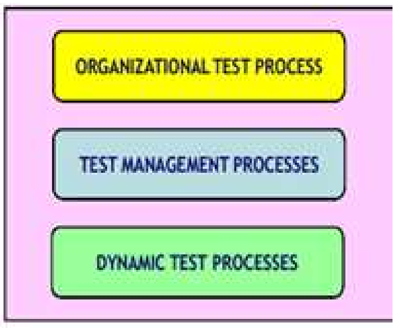

Fig 2.5:ISO/IEC/IEEE 29119 – Test Processes

Test process levels are the standard being instantiated for use at different levels. The organizational test process is instantiated twice: once to develop and maintain the organizational test policy and once for the organizational test strategy. The test management processes are instantiated to develop and implement the project test plan, and also used for each subsequent phase or type of testing for which a separate test plan is created. Although test plans developed using ISO/IEC/IEEE 29119 are expected to include consideration of both static and dynamic testing the lowest layer of processes in ISO/IEC/IEEE 29119 is currently limited to dynamic testing. These dynamic test processes would be implemented whenever dynamic testing is required by a test plan (e.g. for unit testing, system testing, performance testing).

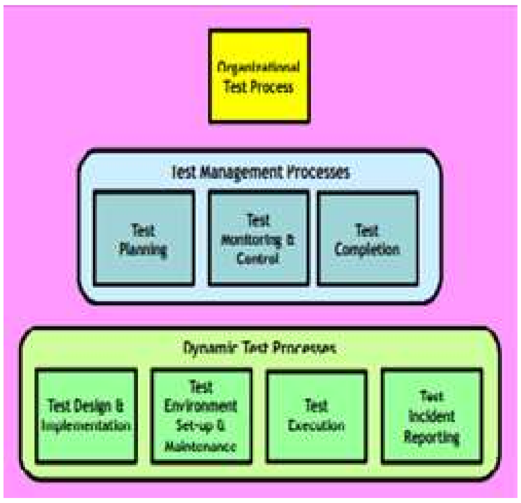
Organizational Test Process
Test Management Processes
Test Planning
Test Monitoring & Control
Test Completion
Dynamic Test Processes
Test Design & Implementation
Test Environment Set-up & Maintenance
Test Execution
Test Incident Reporting

Figure2.6: All eight ISO/IEC/IEEE 29119 Test Processes

2.11 The TMMi Software test Framework

According to the TMMi Foundation the Test Maturity Model Integration (TMMi) has been developed to serve as a guideline and point of reference for test process improvement. (Van, 2009).The sources that served as input to the development of the TMMi was the CMMI, TMM, (Gelperin and Hetzel's, 2004) growth of software testing discussed above. According to (Van et al., 2009), the TMMi can be used as a complementary model to the CMMI.

The TMMi uses a staged approach for process improvement and consists of five maturity levels. An organization must work through all the lower levels of maturity before it can attempt to reach the higher levels. Each maturity level consists of key process areas containing specific and generic goals. The specific and generic goals in each process area must be present to satisfy that process area. Each specific and generic goal is made up of specific and generic practices respectively. All the specific and generic goals of each process area must be reached before the maturity at that level can be attained. This case study will mainly focus on TMMi level 2 and level 3.The first reason for this is that a test process improvement process that aims to reach TMMi level 2 can take up to two years and is thus not a small endeavor (Van, 2008).

The second reason is that at the time of writing, only TMMi levels 2 and 3 were defined by the TMMi Foundation. The definition of TMMi levels 4 and 5 are planned for release in late 2009 or early 2010. Therefore, this research of this study will not focus on a higher level of maturity than TMMi level 3

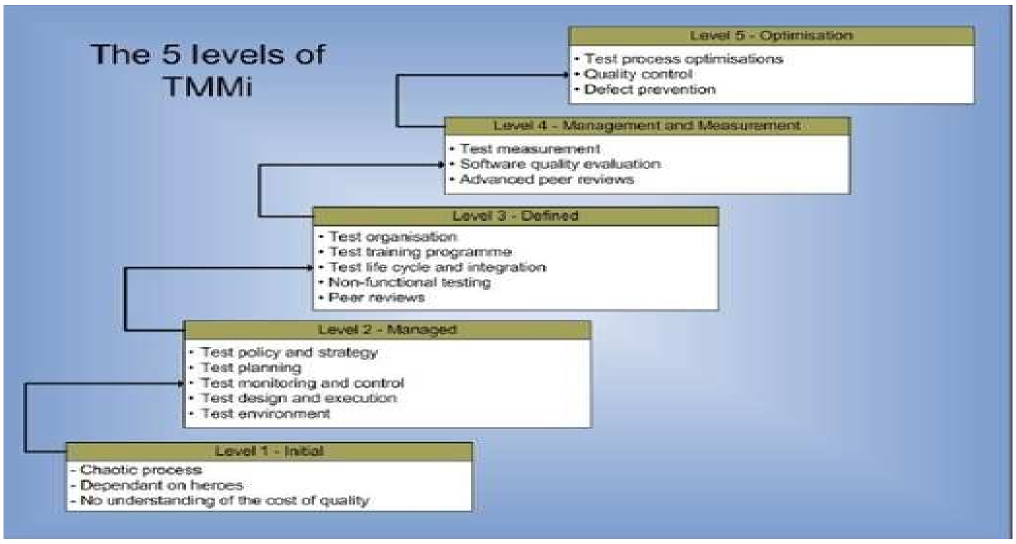

Fig2.7: The TMMi Levels

2.11.1 The TMMi Levels

Level 1 – Initial

TMMi recognizes testing as chaotic with undefined test process and testing is often considered as part of debugging in the organization. Most likely the success in the organization is determined by the heroic actions or the risks are accepted by the customers and users. The organizations are often characterized by over commitment, abandoning of processes in crisis times, and an inability to repeat the successes. There are no key processes involved at this level and is highly recommended for the advancement into next level.

Level 2 – Managed

Testing is a managed process that clearly separates it from debugging and helps to ensure that the existing practices are retained during times of stress. The main objective of testing is to verify that the product satisfies its requirements. However, testing is still perceived by many stakeholders as being a project phase that follows right after coding. In this level testing is recognized as multileveled ranging from unit to acceptance test. For each identified test level there are specific objectives defined in the organization-wide or program-wide test strategy. The process areas at level 2 are:

- Test Policy and Test Strategy
- Test Planning
- Test Monitoring and Control
- Test Design and Execution
- Test Environment

Level 3 – Defined

At this level, organizations understand the importance of reviews in quality control and implement a formal review program linked to dynamic test process. Testing is fully integrated into the development lifecycle and the associated milestones. At this level test process improvement is fully institutionalized as part of the test organization’s accepted practices and testing is perceived as a profession. The process areas at level 3 are:

- Test Organization
- Test Training Program
- Test Lifecycle and Integration
- Non-Functional Testing
- Peer Reviews

Level 4 – Measured

At this level testing becomes a measured process for the implementation of the Level 2 and Level 3 process areas to encourage further growth and accomplishment of the test organization. Testing is perceived as evaluation that consists of all testing lifecycle activities concerned with validation and verification for a product or related work products. With respect to product quality, the presence of a measurement program allows an organization to implement a product quality evaluation process by defining quality needs, quality attributes and quality metrics. Products or related work products are evaluated using quantitative criteria for quality attributes such as reliability, usability and maintainability.

Level 4 also covers establishing a coordinated test approach between peer reviews of static testing and dynamic testing and the usage of peer reviews results and data to optimize the test approach with both aiming at making test more effective and more efficient. Peer reviews are directly integrated with dynamic testing process and is a part of the test strategy, test plan and test approach. The process areas at level 4 are:

- Test Measurement
- Product Quality Evaluation
- Advanced Peer Reviews

Level 5 – Optimization

At level 5, an organization is capable of continuously improving its processes based on a quantitative understanding of statistically controlled processes. Improving test process performance is carried out through incremental and innovative process and technological improvements. The testing methods and techniques are optimized and there is continuous focus on fine-tuning and process improvement. The defect prevention process area is established to identify and analyze common causes of defects across development lifecycle and define actions to prevent similar defects from occurring in the future. Test process optimization process area introduces mechanisms to fine-tune and continuously improve testing.

The process areas at level 5 are:

- Defect Prevention
- Quality Control
- Test Process Optimization

To summarize, TMMi process areas provide wide support and a more detailed specification of what is required to establish a defined verification and validation process. TMMi framework addresses all test levels (including static and dynamic testing) of structured testing (test lifecycle, techniques, infrastructure and test organization). TMMi provides an excellent reference model to aid in test process improvement for both internal and external customers and suppliers. (TMMi foundation, 2012)

2.12. Review of Related Literature

Ng.et.al (2004) conducted a study on Software Testing Practices in Australia. The survey focused on five major aspects of software testing, namely testing methodologies and techniques, automated testing tools, software testing metrics, testing standards, and software testing training and education. The objective of the study was to determine the types of testing techniques, tools, metrics and standards that organizations in Australia use when carrying out software testing activities with the purpose of providing a concise picture of the current industry best practices. Both qualitative and quantitative methods were used. They identified the limitation, strength and weakness in software industries and the major attributes of software testing. They also identified the optimum relationship between testing and software quality; to ensure that testing strategies are in place which yield the highest quality software.

Lalit and Joel (2015) conducted a survey on the state of the art in software testing practices of different countries across the world with the objective of finding a new fact and trends in software testing. The study used a survey method through online designed questionnaires.

The major findings indicated that there are changes in terms of commitment of tester, resource allocation, availability of training for tester and their feelings and the stable environment created in their job position. They also observed that there are still challenges in terms of time allocation for testing, skill gap, resource allocation, standardization and use of different techniques across world. Garousi and Varma (2010) conducted a survey on changes in software testing practices in the Canadian province of Alberta from 2004 to 2009 in terms of testing tool usages , test techniques

usage, level of test automation, test framework selection, test effort and team formation. By deploying both qualitative and quantitative methods they identified that Alberta companies still face approximately the same software engineering economic issues as do companies in other jurisdictions. Compared to 2004, more companies are spending more effort on pre-release testing.

More organizations are using coverage analysis to terminate testing. But still informal criteria are used often. Cost and lack of expertise are two major barriers for adaptation of testing methodology and tools.

Lee et.al, (2012) conducted a survey on software testing practices in Korea constituting a wide variety of companies and experts that are involved in software testing. The aim is to identify the current practices and opportunities for improvement of Software testing Methods and tools (STMTs). The survey results revealed five important findings regarding the current practices: STMTs and opportunities for improvement: low usage rate of STMTs, difficulties due to lack of STMTs, use of testing tools in a limited manner, demand for interoperability support between methods and tools of software development and testing, and need for guidance to evaluate STMTs or to describe the capabilities of STMTs. They have also mentioned that these findings and other related results from the survey will be useful for improving STMTs practices and developing software testing tools.

Adnan et al., (2010) conducted an industrial survey on Contemporary Aspects of Software Testing. The study focused on current practices and preferences on contemporary aspects of software testing based on perceptions of different categories of respondents about software testing process. The study finally identified that there were notable discrepancies between preferred and actual testing practices. Their recommendation includes continued efforts to provide guidelines in the adaptation of the testing process to take care of these discrepancies and thus, improving the quality and efficiency of the software development and testing.

Saraf (2016), conducted an Investigation of the use of test automation in software quality assurance in Norwegian companies and organizations with the aim of identifying the practice of test automation. Using mixed research approach the researcher found that the dominance of manual testing over automated testing, testers' bad feeling in conducting testing, allocation of lesser time for testing by company owners and lack of budget in purchase of relevant test tools were the major problems. They recommended the use of test tools as inevitable in Norwegian companies.

Kapur et al. (2014) conducted research on Measuring Software Testing Efficiency Using Two Way Assessment Technique with the aim of optimizing test techniques using problem conceptualization as a method. They have identified lack of resource like finance, time, and lack of experts as a challenge they recommended two way means of achieving the efficiency and effectiveness, i.e., through working in ideal situation where there exists resource and worst situation where lack of resource exists.

Javed et al., (2012) conducted a study on how to improve software quality assurance in developing countries. The objective of the study was identifying some the major problems associated in quality assurance and propose a solution on how to mitigate the problem. By applying qualitative and quantitative method they identified lack of experts, poor communication, poor documentation, and finance as major problems. They proposed solutions including creating and adopting the CMMi model, use of certified test specialist, motivating test team to change their attitude, avoiding internal politics, griping in the domain knowledge and using of simulation.

Nirmala et al., (2013) have conducted research on automated test framework for software quality assurance. The objective was generating test cases automatically and to decrease the cost of testing in addition to saving the time of deriving test cases manually. The ultimate goal is improving quality. The study finally came up with an automated test framework that generates the test cases automatically, evaluates those test cases and produces the test summary report as well as software quality assurance report. The new framework was designed for performing both functional and non functional testing.

Beer and Rudolf, (2013) conducted study on the role of experience in software testing practice. The proposition was that experience plays a major role in software testing and it is an important factor for developing test cases. The proposition relates to the reported evidence of benefits of experience-

based testing. By applying mixed method research approach the authors found that although the development of testing knowledge was an important aspect, substantial domain knowledge was also required for testing, which could only be developed adequately by working in the domain or by longterm involvement in a project. According to the authors, the typical path of knowledge development of senior testers started with domain knowledge.

Testing knowledge was developed later while working as tester and attending additional seminars. Advanced testing was usually introduced by external consultants working together with domain experts. The authors recommended finding the optimal mix of testing knowledge and domain knowledge as a vital issue for successful projects and a major task for future research. The following specific recommendations were also made by the authors:

- Authors identified that there are difficulties in specifying requirements consistently, completely and correctly. For this challenge they recommended reviewing and improving requirements specifications as an effective measure to improve testing. Investing on applying experience-based testing was also mentioned as a solution to overcome issues in imperfect specifications.
- They also recommended that the tools and techniques that support experience-based testing should be designed in such a way that it can 1) directly support the incorporation of the tester's experience, e.g. as additional source for generating test cases. 2) foster gaining and sharing new experience throughout testing activities in addition to producing tests results.

Anitha (2013) conducted a study on a brief overview of software testing techniques and metrics. The finding of the study indicated that the software testing can be very costly and recommended automation as a good way to cut down time and cost.

Summary

As it is indicated in the review of related works studies mostly focused on surveying the existing software testing practices by considering testing methodologies, techniques, tools, metrics, test automation, test effort, test team, etc.; Areas of challenges identified by the studies include – lack of software testing methods and tools, lack of budget, high cost of testing, lack of commitment of testers, poor resource allocation, lack of training for testers, less time allocation, skill gap/lack of expertise, lack of standardization, low level of usage of methods and tools, the dominance of manual testing over automated testing, testers’ bad feeling in conducting testing and lack of commitment, poor communication and poor documentation.

Proposed solutions include – the need to provide guidelines in the adaptation of testing process, creating and adopting CMMi model, use of certified test specialist, motivating test team to change their attitude, avoiding internal politics, griping in the domain knowledge, using simulation, applying automated test framework that generates test cases automatically, investing on experience-based testing, developing testing tools that can incorporate tester’s experiences, reviewing and improving requirements specifications and automation as a means to cut down time and cost.

Although prior studies identified different challenges and proposed solutions, non of them have dealt with developing software testing framework that can guide software testers towards conducting effective and efficient software testing process in a resource constrained environment.

CHAPTER THREE

RESEARCH DESIGN AND METHODOLOGY

This study has three major aims: assessing the existing software testing practices, identifying major challenges associated with software testing and proposing a framework that guides software companies in conducting effective software testing given a resource constrained environment. This chapter presents the methodology that the researchers followed to achieve the above stated aims. The chapter constitutes discussions on research design, sampling, data collection and data analysis aspects of the research methodology.

3.1 Research design

A mixed method research approach that combines both quantitative and qualitative research methods (Creswell et al., 2011) was applied to address issues embedded in the three major aims of the study, which are mentioned in the preceding paragraph. Proponents of mixed methods research appreciate the value of both quantitative and qualitative worldviews to develop a deep understanding of a phenomenon of interest (Venkatesh, et al., 2013).

In this mixed approach evidences are mixed and knowledge is increased in a more meaningful manner than either model could achieve alone(Creswell & Plano, 2007). We found this approach more relevant for addressing the three aims of our study. The qualitative method enabled us to gain real-life contextual understandings and multi-level perspectives (Creswell et al., 2011) on software testing practices and associated challenges in the context of Ethiopian Software industry. The researchers employed quantitative research for the purpose of assessing the magnitude and frequency of using software testing methods, tools and techniques in the software companies. Therefore, the researchers used both qualitative and quantitative data to understand a research problem (Venkatesh et al., 2013).

3.2 Sampling

According to Addis Ababa trade and economic development office report more than 400 business licenses were taken from the Ministry to establish Software Company. But most of the individuals or groups are engaged in other related activities including sell of computer hardware and related components, network devices as well as providing network installation and maintenance services. Therefore, the researchers applied a non-probability purposive sampling technique to identify ten recognized and active software development companies, their project managers and employees. The main goal of purposive sampling is to focus on particular characteristics of a population that are of interest, and also the selection are made their experience, investment level, active involvement in the industry which enable the researchers to extract valuable experience, challenges and opportunities. which will best enable us to answer our research questions. The sample being studied is not representative of the population. The aim of sampling informants with a specific type of knowledge or skill or experience (Li et al.,2006), which in our case is in software testing. The major criteria used to select the companies were active involvement in the software industry and extensive experience in software development. This strategy helped us to collect a more representative view of a population of interest, thus supporting transferability, or the ability to apply findings to the population at large (Krefting, 1991).

From the 10 software companies a total of 15 respondents participated in a one-on-one interview. All of the participants under this category are software project managers. In addition a total of 87 employees participated in the survey responding to the questionnaire.

3.3. Company Profile

As it is indicated in Table 4.1 the 10 companies that were involved in the study were selected based on their size (number of staff it has), company type, infrastructure, technological usage. Hence the companies were divided into large (having more than 50 staff and well equipped), Medium (having staff between 25-50 and well equipped) and small (having staff number less than 25 staff and working with limited resource). The profiles of companies are summarized in the following table.

No	Company Name	Description of the company
1	INSA	Large scale governmental organization having more than 500 staff
2.	Apposite	Private owned medium scale organization having more than 20 staff
3	Castor	Private owned medium scale organization having more than 10 staff
4	Techno brain	Large scale nongovernmental organization having more than 30 staff
5	Cnet	Large scale nongovernmental organization having more than 30 staff
6	GCS	Large scale nongovernmental organization having more than 30 staff
7	Cyber soft	Large scale nongovernmental organization having more than 30 staff
8	TYC	Private owned small scale organization having less than 5 staff
9	Appnova IT Solution	Private owned small scale organization having less than 5 staff
10	Sol Net	Private owned small scale organization having less than 5 staff

Table 3.1 Company Profile

3.3 Data Collection

Both qualitative and quantitative primary data were collected. An in-depth semi-structured interview technique was used in order to get valuable insights into the existing practices of software testing and major challenges faced by software companies in Ethiopia. Interviews were chosen to collect qualitative data because the format allowed for significant probing vis-à-vis a two-way communication that provided in-depth descriptions of topics being discussed. The interview protocol was developed that included series of questions pertaining to the respondent's experiences and practices in software testing. The questions mainly focused on the existing software testing processes, methods, tools and techniques; policies, strategies and plans pertaining to software testing; the strengths and weaknesses in software testing; major challenges faced and mechanisms used to cope up with the major challenges. Probing prompts were used to collect more in-depth information for responses that seem ambiguous or confusing. A total of 15 respondents participated in the interview. All participants are at the position of project manager and quality and assurance officer. Each interview took a minimum of 40minutes hour and a maximum of 1 hour duration. Interviews were recorded with digital recorder with the permission of the interviewee. Assurance was given to the interviewee on the confidentiality of their responses before the interview. Such assurance minimized the digital recording drawbacks. In addition to audio recordings, the researcher kept written notes. Survey method was also deployed to collect quantitative data. An instrument pre-tested and validated by prior studies were adopted and used to collect the data. Collecting quantitative data using questionnaire mainly aimed at measuring the extent and frequency of using software testing methods, tools and techniques in the software companies.

The questions focused on the existing software testing processes; the availability and training level of software testers; time allotted for software testing; type of testing methods, techniques and tools being applied; the extent of use of Software testing standards; challenges in adoption of software testing methods, tools, techniques and test automation and possible remedial actions to cope up with the challenges. A total of 103 questionnaires were distributed to respondents and 87 questionnaires were returned with a response rate of 84.4%.

3.4 Data Analysis

Interviews in the form of audio recordings were transcribed verbatim. Each response was categorized into themes based on the topics of interview guide. These themes included: software test processes; decision criteria in selecting methods, tools, techniques; strengths, weaknesses & challenges; mechanisms for coping up challenges. Based on this thematic categorization, data were ready for more detailed analysis. Therefore, thematic coding and thematic analysis were used in order to extract major findings that address research questions.

Descriptive quantitative data analysis was used for the survey data using Microsoft Excel (Microsoft, 2010).Manual checks for accuracy of the data entry were made on a randomly sampled 10% of downloaded questionnaire responses. If errors were identified, more checking and correcting was performed. Finally, percentages were calculated and results were summarized using table’s graphs and charts.

3.5 Validation

As it is indicated in the research questions and specific objective, the final output of this study is proposing an integrated software testing framework for a resource constrained environment. The framework was validated.

CHAPTER FOUR

DATA PRESENTAION AND ANALYSIS

4.1 Introduction

Data analysis involves critical thinking. The data analysis is done after collecting all the data from the respondents. Thus, the analysis of the study follows the objective of the research. Moreover, the data gathered through the above-mentioned methods were analyzed using statistical tools, such as graphs, tabulation and percentage using Microsoft Excel. Whereas, the data from interviews and observations were presented to assess the existing software testing practices of Ethiopian software companies. The responses obtained through questionnaires were integrated with interview results and physical observation in order to address the research questions.

4.2. Presentation & Analysis of Data

Respondents' Demographic Data

As it is presented in table 4.1 the dominant number of respondents are male (75%), within the age limit of 20 – 30 (52%) and married (58%). More than 64% of them have educational level of first degree and above.

Variables	Proxies	Total Number	Percentages
Sex	Male	66	75%
	Female	11	12%
	No Response	10	11%
	Total	87	100%
Age	20 – 30	45	52%
	31 – 40	17	31%
	Above 40	0	0
	No Response	15	17%
	Total	87	100%
Marital Status	Married	51	58%
	Not Married	23	39%
	No Response	13	3%
	Total	87	100%
Educational Status	Diploma or below	15	19%
	First degree	49	64%
	Second degree and above	13	17%
	Total	87	100%

Table 4.1 Respondents Demographic Data

Current position

As shown in Figure 4.1 below the largest proportion of respondents (49%) are working as software programmer or developer followed by software tester (11.49%). The rest of the respondents are engaged as requirement engineer and project manager that account for 10.34% and 5% respectively. From this data it is clear that the number of testers is limited as compared to software developers. It also signifies that less attention is given to software testing process. This conclusion is supported by the data from the interview. One of the respondents said that:

> "*Companies lack skilled manpower to conduct a sufficient test activity. Most of the testing activities are done by developer during the development phase with less attention and limited time*"

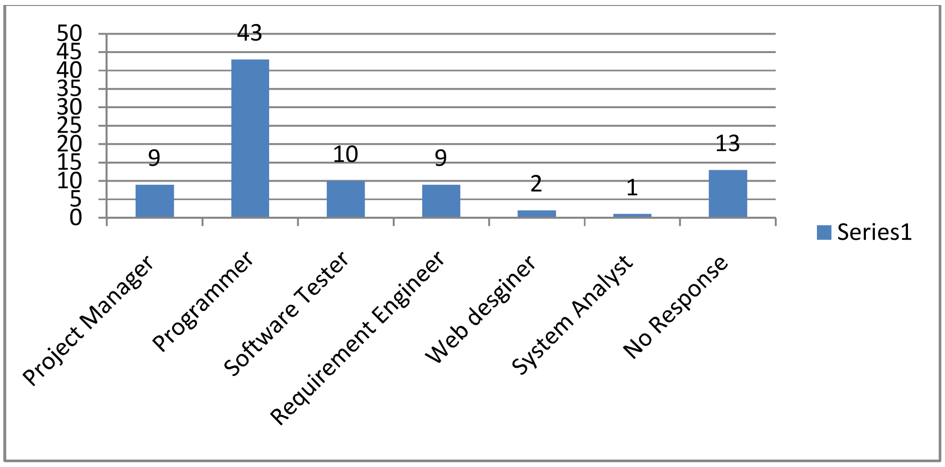

Fig. 4.1: Position of respondents

Experience

As it is presented in Figure 4.2 the majority (53%) of the respondents have less than one year of experience which is followed by respondents with experience of 1 – 3 years (14%) and five years and above (10%).

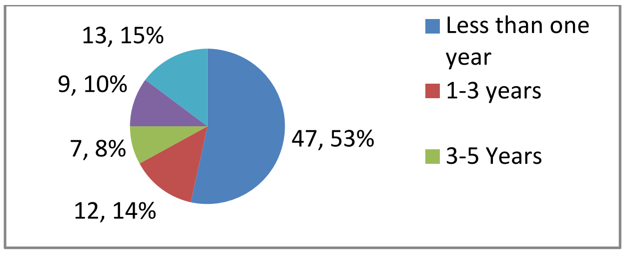

Fig. 4.2: Experience of Respondents

According to Beer and Rudolf (2013) experience-based testing plays a major role in performing efficient and effective software testing. From this perspective our software companies do not have experienced testers which has a negative impact on the testing process and quality of the software product.

Software Testing Process

Respondents were asked whether they apply software test their organization. Most companies (47%) don't have test process where as 29.80% of the companies perform software testing.

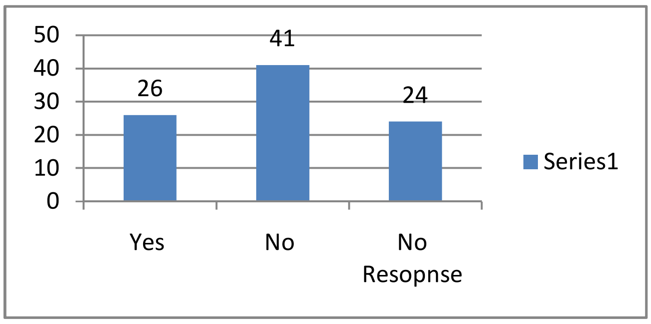

Fig. 4.3 Availability of test process

Primary Responsibility for Software Testing

Respondents were asked to indicate the existence of department or staff responsible for software testing in their company. The result shown in Fig. 4.4 signifies that 57.47% of the respondents stated that there is no formal test staff or department in their company. Whereas 34.48% of them confirmed that their testing staff is divided among application groups and thus there is no single, centralized software testing department. One of the interview participants also mentioned that:

> *"Most companies lack formal and centralized structure which is responsible for performing software testing."*

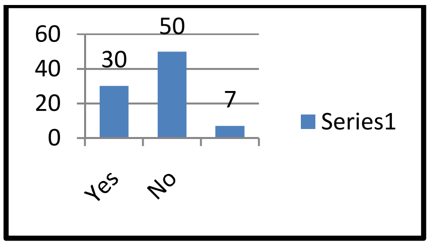

Fig 4.4 Test Responsibility Level

Training on Software Testing

Respondents were asked whether they get formal training in software testing. The result indicated in Fig. 4.5 shows that 80 % of respondents didn't get formal training on software testing whereas 20% of them get training. Based on the interview result we learned that the training provided is more of in-house which is given by senior staff who have a better experience and background in software development and testing. Regarding the problem related to training one of the interview participants mentioned that:

"Lack of training is one of the major problems in our company in the areas of software testing. The company owners do not invest on the provision of relevant training"

Black (2008) also confirmed that lack of training has a negative impact on quality of software.

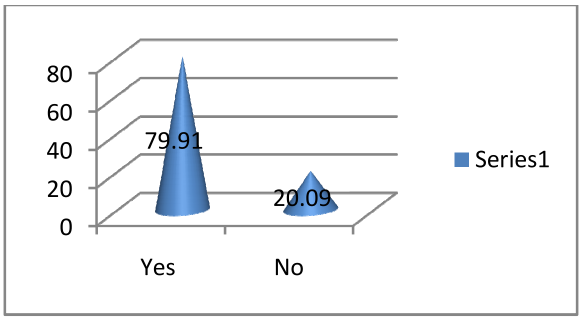

Fig. 4.5 Training on Software Testing

Practice of Major Test Process

Respondents were asked to indicate, which test process they perform and the kind of tools they use if any. As it is indicated in table 4.2 the majority of the respondents indicated that they don't perform test processes like test planning, estimation, design, execution and management. Whereas 66% of the respondents agreed that they apply test reporting.

Test Process Type	Yes		No	
	Freq.	%	Freq.	%
Test Planning	23	20	64	55
Test Estimation	16	13	71	61
Test Design	17	14	70	60
Test Execution	16	13	71	61
Test Management	13	11	74	64
Test Reporting	76	66	11	9

Table. 4.2 Practice of Major Test Process

Proportion of Time Allotted for Software Testing

Respondents were asked to indicate the proportion of time spent on software testing. The majority of respondents 42% confirmed that the time allotted for software testing are between 26 – 50%. Those who allot less than 25% of the time accounted for 19%. As one of the interview participant confirmed "*the time spent on software testing ranges between 20% to 30% of the total software development time.*" But other studies indicated that at least 50 % of the time should be allotted for software testing (Bartolena, 2007). Therefore, in the context of Ethiopian software industries very limited time was allotted for software testing. Studies conducted by Javed, et al. (2012) also reported the same problem.

Time Spent	Freq.	%
76%-100%	5	4
51%-75%	12	10
26%-50%	48	42
Less than 25%	22	19

Table 4.3 Time spent on testing

Frequency of Use of Different Software Testing Methods

Respondents were asked to indicate the frequency of use of different software testing methods including Review, Inspection, Test automation and manual testing. As show in the table below the majority the respondents confirmed that review (56%), inspection (47%) and manual testing (43%) methods are being used sometimes. But still higher number of respondents (33%) said that they use manual testing always. The majority of the respondents indicated that they don't use automated testing tool.

	Always		Sometimes		Never		N/A		Unknown	
Methods	**Freq.**	**%**	**Freq.**	**%**	**Freq.**	**%**	**Freq.**	**%**	**Freq.**	**%**
Review	17	20	49	56	0	0	0	0	21	24
Inspection	0	0	41	47	0	0	0	0	0	0
Automated Testing	0	0	17	20	28	32	20	23	22	25
Manual Testing	29	33	37	43	0	0	0	0	21	24

Table 4.4 Frequency of Use of Different Software Testing Methods

Test methods /tools/ Techniques Utilization Level

Respondents were asked to indicate the level of utilization of different software testing methods/types, Techniques and Tools. As it is show in table 4.5 below the majority of the respondents strongly agreed that they apply unit test and integration test with mean value of 4.5 and 4.6 respectively. In the case of techniques most respondents agreed that they apply white box techniques with mean value of 2.9. Regarding tools, the majority of respondents strongly agreed that they use open source tools with mean value of 4.1. In their study Lee et al. (2012) also found that although the use of software methods and tools gives opportunities for improvements, low usage rate of software test methods and tools were observed due to cost of tools and methods. From this we can conclude that most software companies dominantly practice functional level testing and they don't give much attention to system test and non-functional testing including security and performance which highly compromises the quality of the software. In terms of tools only depending on open source test tools and ignoring the commercial tool has its own negative impact on efficiency and effectiveness of software testing and thus, the quality of software. One respondent recommended that:

"All companies should at least perform unit testing, integration testing and performance testing since they are fundamental test types in spite of the resource constraints that companies face."

	Total	Mean	Stan.Dev
1.Method/Type			
1.1 Unit Test	87	4.5	1.26
1.2 Integration Test	87	4.6	1.29
1.3 Security Test	87	2.6	0.73
1.4 Performance Test	87	2.2	0.61
1.3 System Test	87	2.5	0.7
2.Techniques			
2.1.Black box test	87	1.7	0.47
2.2.White box test	87	2.9	0.8
3.Tools			
3.1.Open source tools	87	4.1	1.15
3.2.Coomercial Tools	87	1.8	0.5
3.3.Locally developed tools	87	0.7	0.29

Table 4.5 Test methods /tools/ Techniques level of Utilization

In relation to the above question respondents were asked to list the type of tools they use for each level of testing. Some of the tools include Selenium (for Unit Test), Jmeter (for Integration Test), Vpscan (for System Test), Sanipro (for Security Test) and Test link (for Performance Test).

Usage of Automated Test Tools

Respondents were asked to indicate whether they use automated software test tools. As it is shown in Fig. 4.6 below only 10% of the respondents confirmed that they use automated test tool whereas 54% of them responded that they don't use automated test tools. Although authors like Anitha (2013) argue that automation is a good way to cut down time and cost, software companies in Ethiopian context couldn't maximize such benefit since the level of utilization of automated test tools is very low.

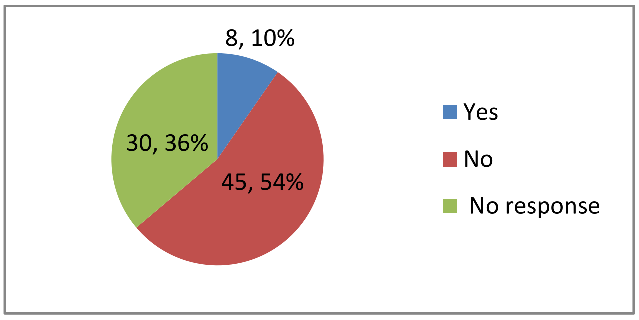

Figure 4.6 Usages of Automated Test Tools

Use of Test Tool for Test Processes

Respondents were asked to answer for which test process do they apply test tool. As indicated in the table below companies use tools mainly for test reporting which accounted for 57% of the total respondents. Regarding test design and test management the majority of respondents, 61% & 60% respectively, confirmed that tools are not used in the stated two test processes.

Test process	**Yes**		**No**		No response	
	Freq.	**%**	**Freq.**	**%**	Freq.	**%**
Test Design	23	26	53	61	11	13
Test Management	26	30	52	60	9	10
Test Reporting	50	57	30	34	7	8

Table 4.6 Use of Test Tool for Test Processes

Software Testing Standard

Respondents were asked to indicate which standard they are applying in their company. As we learn from Figure 4.7 About 49 respondents (67.12%) confirmed that they don't apply any standard at all to conduct software testing followed by 4 (4.6%) that use standards like CMMi and TMMi. One of the participants of the interview said that:

"Most of the companies in Ethiopia don't strictly follow the international standards or don't have customized standards to carry out testing"

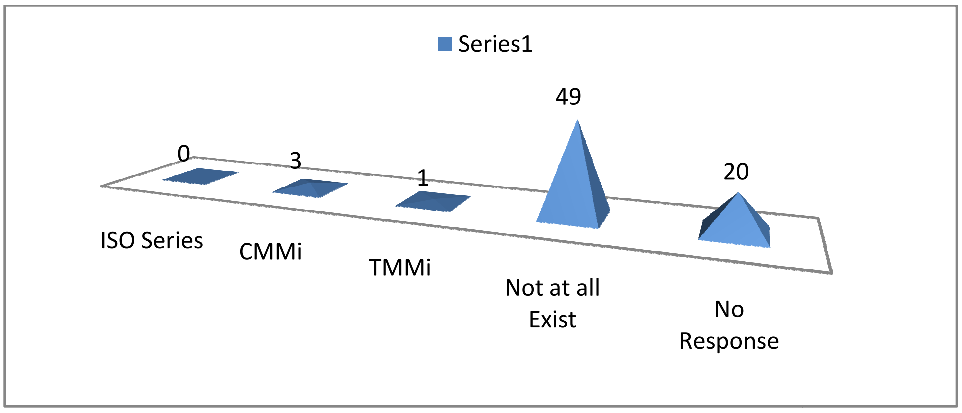

Figure 4.7 Software Testing Standard

Customer participation

Respondents were asked whether their company encourages customers to participate in software testing. As indicated in Figure 4.8 below 72% of respondents confirmed that they don't encourage customer participation during the test process whereas 28% of them said that they encourages customer to participate in testing activities. One of the interview participants said that:

"Our company allows the participation of customers during test process and we believe that their involvement can assist us in finding bugs during the test process by comparing the actual system against the minimum expectation represented by the system requirements."

Other participant also added:

"Customer is a key factor in the success of software industry; ….our company provides relevant training until they can operate on the system by themselves. We also believe that unless the customer has the right knowledge the industry will not grow as it is expected ".

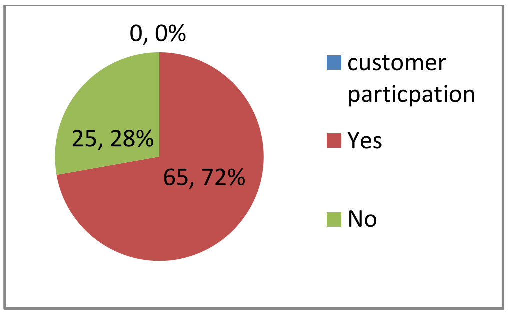

Figure 4.8 Customer participation

Decision Criteria for Selecting Test Level

Respondents were asked to indicate their decision criteria for the selection of test level (unit, integration, system….). As it is presented in the following table respondents almost strongly agreed that all the decision criteria are valid and used as a basis for selecting the test level.

	Total	Mean	Stan.Dev
Availability of Expertise	87	4.32	1.31
Experience of testers	87	4.25	1.35
Availability of adequate budget	87	4.14	1.25
Complexity of the tested system	87	4.08	1.23
Delivery Time	87	4.14	1.25

Table 4.7 Criteria for selecting test level

Challenges in the Use of Tools, Methods, Techniques and Test Automation

Respondents were asked to rate their agreement on the possible challenges that affect the adoption of methods, techniques, tools/automation. As it is shown in Table 4.8 all challenges except lack of proper attention and reservation in usefulness and cost effectiveness affect the adoption of test methods and techniques. But the dominant factors that most respondents agreed are lack of expertise and lack of adequate budget. In addition, all the listed challenges affect the use of tools including test automation, but the dominant factors are lack of adequate budget and time-consuming to use.

Challenges	**Total**	Methods/Techniques		Tools/Automation	
		Mean	Stan.Dev	Mean	Stan.Dev
Lack of expertise	87	3.87	1.04	3.67	0.98
Lack of adequate budget	87	3.87	0.04	4.06	1.09
Costly to use	87	3	0.8	3.19	0.85
Difficult to use	87	3.5	0.94	3.55	0.95
Time-consuming to use	87	3.49	0.93	4.14	1.11
Lack of proper attention	87	1.14	0.33	3.88	1.04
I don’t think it is useful or cost effective	87	2.49	0.67	3.56	0.95

Table 4.8. Challenges in Tools, Methods and Techniques and Tools Automation

Recommended Solutions for Challenges in Software Testing

Respondents were asked to identify the possible solutions for challenges in software testing in general. As we learn from the following Table 4.9 all the solutions are strongly supported by the respondents with slight variations in the rating.

Solutions	Total	Mean	Stan.Dev.
Recruit appropriate skilled testers	87	4.41	1.47
Training existing testers	87	4.56	1.61
Identify and purchase the right testing tools	87	4.08	1.35
Identify new methods and techniques appropriate for our context	87	4	1.23
Developing and adopting appropriate testing process model	87	4.2	1.4
Allocate more budget and resource for software test process	87	4.25	1.41
Engage with specialist test provider to benefit from External expertise to maximize the level of importance and benefit	87	4.28	1.43

Table 4.9 Recommended solutions for challenges

4.3 Summary of Results from Interview

As it is stated in the methodology, interview method was employed to collect qualitative data. Fifteen software project managers, quality assurance team leaders and senior testers were involved from software companies selected for the study. Interview protocol was developed in such a way that the questions can address the major research objectives. The results of the interview are summarized in themes which are derived from the interview question and questionnaires and observation made during the study using the following table 4.10.

No	Themes	Explanations
1.	Software test process	The following key results were found in relation to the question pertaining to the current software test processes being performed in the company. Respondents confirmed that: • there is no formal software testing process in their company; • company owners give less attention to software testing and give more emphasis for increasing their revenue through fast development and delivery of final software product to customers; • because of the above situation they lack proper awareness and practical skill in software testing process • they perform mainly functional level testing that constitutes 10% of the entire development process • limited skilled staff, limited time, lack of budget and technological infrastructure are the major constraints • In addition to the respondents that stated about lack of test process in their company, there are also few respondents from only three companies that stated about their good practices in relation to software testing. They said that: • their company practically applies the major test process like test planning, designing, executing, reporting and management of the whole test process (Techno Brain, Aposit, CNet, INSA); • they have dedicated test team (INSA, Techno Brain, Aposit, CNet) • their company is certified in CMMi 4 and apply the international standard in software testing process (Techno Brain) • their company uses commercial software test tools (INSA) • their company uses open source software test tools (almost all companies)
2.	Test policy, Test strategy and Test plan	Regarding test policy, test strategy and test plan the following results were found based on the researcher's observation and responses from participants: • most of the companies include in this study do not have test policy, strategy and plan;

		• only few companies are in a better position in terms of developing their own policy, strategy and test plan. But it is not yet operationalised;
3.	Best practices	Some of the best practices mentioned by respondents include: • assigning quality assurance personnel who is primarily responsible for testing related activities in some of the medium scale software companies have started, although the application of standard software testing practice is at its lowest stage • introducing internal audit system (Techno Brain) • having well established internal and external training program and testing center which is strongly committed to the transfer of knowledge to internal staff and customers (Techno Brain)
4.	Challenges	The major challenges identified by respondents include: • lack of sufficient budget, • lack of skilled manpower, • lack of training • lack of appropriate test environment including internet connectivity, workspace, guidelines, • lack of awareness on test processes, methods, techniques and tools, • information gap among staff members, • poor documentation of test processes, • tight schedule being assigned for testing, • being unable to adopt test standards • developers serving as software testers • demotivation among staff of software companies • lack of collaboration among project managers, developers and testers • dynamism and fast changing software environment and being unable to adjust to such situation
5.	Proposed Solutions	Participants of the interview proposed the following solutions in order to cope up with challenges: • raising organizational commitment towards software testing, • placing internal audit system, • providing relevant training/creating awareness among staff members regarding software testing processes, methods, tools, techniques, • restructuring and institutionalizing software testing as an independent function, • recruiting personnel with adequate skill in software testing coupled with motivational schemes, • experience sharing in software testing practices internally with

		senior experts and externally with software companies, • developing or adopting customized standards for software testing which can easily be applied in our context, • enforcement by the government towards ensuring software quality through applying acceptable standards, • provision of adequate infrastructure with optimal utilization through pull system, • the current situation doesn't allow small scale software companies to produce quality software, survive in the market and transform themselves into medium scale. Therefore, they need government support and protection through creating fair competitive environment, • proper documentation of test processes including test cases and test report, • create a platform for communication among staff and with customers, • giving due emphasis on the fulfillment of each and every requirement specification during testing, • encouraging customers to participate in software testing

Table 4.10 Summary of interview results

CHAPTER FIVE

Software Testing Improvement Framework (STIF)

5.1. Background

A framework is a constructive blend of various strategies, standards, perceptions, methods, conventions, system hierarchies, modularity, etc. which are structured to represent an industry process (Nirmala and Maheswari, 2015), in our case software testing process. One of the primary objectives of software engineering is delivering high quality software to the customer (Nirmala and Maheswari, 2015). Software testing plays a key role in this regard.Testing takes a large share of software development efforts (Karlstrom et al., 2005).

Although different software testing improvement frameworks were developed including TMMi and CMMI they are too expensive for small and medium-sized software companies (Karlstrom et al., 2005) which are dominant in developing nation like Ethiopia. The high level of formalism of the frameworks makes them difficult to introduce in a small and medium sized organization. They also require huge resources which are unaffordable by the existing companies. Companies also lack staff with the required knowledge and skill (Karlstrom et al., 2005).

Therefore, the standards are on a more strategic level, and there is a need for more practice-oriented support (Karlstrom et al., 2005). In order to cope up with such challenges different frameworks were proposed by different authors that address problems linked with specific context and based on requirements of the specific study context. For example, Karlstrom et al., (2005) proposed Minimal Test Practice Framework (MTPF) which defines the kind of practices that are needed in small and emerging software companies but the study focuses on the specific context without addressing the phase based approach to mitigate the problem related to resources .

5.2. Software Testing Improvement Framework

The building of the proposed software test improvement framework starts by identifying the summary of major challenges, practices and opportunities of the Ethiopian software industries so as to mitigate the problems associated to software testing in order to make the testing process efficient and effective by developing a guide line for resource constrained environment like Ethiopian software industries.

5.2.1 Summary of Challenges

The following table 5.1 summarizes major challenges derived from the empirical study. These challenges were the basis for proposing a framework that introduces a phased approach of introducing software testing practices in the Ethiopian software company's context. The major areas identified in this summary are derived both from qualitative and quantitative study.

Areas	**Challenges**
Staff	limited experience of testers
	limited skilled staff/ lack of skilled manpower
	no formal training in software testing
	lack of proper awareness and practical skill in software testing process
	lack of awareness on test processes, methods, techniques and tools,
	developers serve as software testers
	demotivation among staff of software companies
Test process	limited application of test processes including planning, estimation, design, execution and management
	high dependence on manual testing and limited use of automated testing
	The majority of the companies perform unit test and ignore integration, system, as well as non-functional testing (e.g. security, performance, etc.)
	high dependence on open source test tools over commercial tools
	only few of the companies use automated test tools
	Most of the companies do not follow software testing standards
	challenges indicated by respondents in using methods, techniques, tools – lack of expertise, lack of adequate budget, believing that it is costly to use, difficult to use, time consuming to use and poor attention,
	there is no formal software testing process in most of the companies
	they perform mainly functional level testing that constitutes
Organizational Structure	In most of the companies, there is no formal test staff or department
Communication	lack of collaboration among project managers, developers and testers
	There is information gap among staff members
Time	tight schedule being assigned for testing,
Test environment	lack of appropriate test environment including internet connectivity, workspace, guidelines,

Customer	There is no customer participation in software testing in most of the companies
Organizational Commitment	company owners give less attention to software testing and give more emphasis for increasing their revenue through fast development and delivery of final software product to customers;
Budget	lack of budget allocated for testing
Policy/Strategy/plan	No test policy, strategy and plan in most of the companies
Documentation	poor documentation of test processes,
Standards	Most of the companies do not follow software testing standards
Requirement Specification	Requirements are not used as a basis for testing

Table 5.1 Summary of major challenges derived from the empirical study

5.2.2 The Structure of the Framework

Our proposed framework is specifically developed based on the current status of software industries in Ethiopia, the types of challenges they are facing and the requirements they have in terms of improving software testing process. Our framework mainly focused on addressing the existing challenges that software companies face in software testing and possible remedial solutions derived from empirical research and literature. Therefore, based on the existing situation of software industries in Ethiopia and the requirements identified through both quantitative and qualitative study we proposed Software Testing Improvement Framework (STIF). The framework is structured in three major areas of challenge with four sub-categories leveled in three phases. The three major areas of challenge and the corresponding sub categories include:

- **Test Management**: this category encompasses all managerial activities related to software testing. The major challenges in this category are associated with test policy/strategy/plan, departmentalization of software testing, budgeting, staffing and collaborative environment (communication & coordination).
- **Test Environment & Process:** this category has four sub-categories.
    - ***Methods/Techniques***: challenges associated with the proper adoption of test levels (unit test, integration test, system test, security test, performance test, etc.); test techniques (black box, white box, gray box testing).

- ▪ ***Tools***: challenges associated with the use of different software testing tools (open source, commercial or locally developed software testing tools; availability of proper infrastructure – Software, Hardware, Connectivity)
- ▪ ***Test design and execution:*** challenges associated with capturing all the test suites and executing the test
- ▪ ***Test Documentation:*** challenges associated with all sorts of documentations – test case, test report, etc.

- **Standards:** challenges associated with the adoption of different international standards for software testing and quality assurance. The three phases include:
- **Phase I –** Software testing initiation phase (baseline activities – paper works, awareness creation, training.
- **Phase II –**Implementation with minimalist approach
- **Phase III** – Full-scale implementation with advanced feature.

Phases of Implementation	**Test Management**	**Methods/ Techniques**	**Tools**	**Test Design & Execution**	**Test Documentation**	**Standards**
		Test Environment & Process				
Phase 3	• Mgt. Commitment • Staff Motivating • Collaborative Environment • Test monitoring and control	• Optimization of Methods& Techniques	• Commercial test tools	• Comprehensive test execution and review • validation and verification	• Collaborative document authoring and sharing system	• Adoption of international standards • Certification in international standards • Enforcement
Phase 2	• Departmentalization • Budgeting • Recruitment • Training/ awareness • Team formation	• Cost minimizing Methods & techniques	• Open source test tools • Infrastructure	• selection of test case design techniques • defining test case • develop test procedures • basic test execution	• Open source document management system	• Customized standards
Phase 1	• Test Policy/ Strategy/ Plan • Roles and Responsibilities	• Training/ Awareness • Test method/ technique selection criteria	• Training/ Awareness • Define test level • Test tool selection criteria	• Training/ Awareness • Gather requirements specification • Define the test objectives	• Training/ Awareness • Identifying major documents • Organizing documents	• Training/ Awareness • Checklist

Challenges

Fig. 5.1 Software Testing Improvement Framework (STIF) for Resource Constrained Environment

5.2.2 Description of the Framework

I. Test Management

- ***Phase 1:*** The software company should start with defining test policy that describing the principles, approach and major objectives regarding testing. Similarly it should also define test strategy consisting description of the test levels and the testing within those levels.
- The third element is test planning that constitute estimation of the number of test cases, resource utilization, responsibilities, risk and priorities. Another important function under this phase is defining responsibility testers which includes developing a test plan for each new project, administering the test environment, administering the problem reporting, continuously assessing the testing practices and monitoring the need
- ***Phase 2:*** the company should be able to departmentalize test activities with its own staff of testers in order to make software testing independent and introduce responsibility and accountability. Budgeting is also another important activity at this stage which requires a proper and cost effective allocation of testing related budget by considering each requirement, time, expertise, software size and nature. The recruitment process for software test should be based on having skilled personnel. The newly recruited test staff should be provided with appropriate training or awareness session in a form of induction. Team formation is another component in this phase which should be based on project size, duration, available testers and available resources in test environment. There is a need to consider interpersonal skills at the time of forming test team.

  Phase 3: building strong commitment of staff and company owners towards software testing is the most important activity in this phase. The management should be committed in terms of providing sufficient and appropriate resources and managing the time pressure, commercial pressure and workload. There is a need to motivate test team by providing different incentive mechanisms including reward testers for finding good quality bugs.

Keep some weekly or monthly competitions such as ‘Bug of the week’ to reward them. This will help to build a successful software test team”.

Setting clear goals and increasing task variety are some of the ways to motivate test team. The company should create a collaborative platform in order to facilitate communication among project managers, developers and testers as well as sharing expertise, skills and experiences among testers. The platform should enable testers to work cohesively together, follow the test processes and deliver the committed piece of work within schedule. Test Monitoring and control is one of the test management aspects that should be dealt at the third phase. The purpose of test monitoring is to give feedback and visibility about test activities. Information to be monitored may be collected manually or automatically and may be used to measure exit criteria, such as coverage. Metrics may also be used to assess progress against the planned schedule and budget. The monitoring and control should involve measuring the amount of work done in test case and test environment preparation, test case execution, defect identification, test coverage, etc.

II.Test environment and Process:

Test environment and process component of the framework constitutes four major classifications, i.e., Methods/Techniques, Tools, Test Design & Execution and Test Documentation. All the four major parts have their own activities to be implemented in three phases. In all the phases provision of training/awareness are common with respect to Methods/Techniques, Tools, Test Design & Execution and Test Documentation. This is important because one of the key challenges associated with software testers in our context is lack of appropriate skill and/or awareness. The key activities to be performed in each phase under each component of test environment and process are summarized as follows:

- **Methods/Techniques:**
    - Phase 1: developing the right selection criteria for selection of test methods and techniques. This should consider cost effectiveness, time, and availability of experts and level importance
    - Phase 2: implementing cost minimizing methods and techniques which are selected based on the criteria set in phase 1.
    - Phase 3: mainly focuses on optimization of test methods and techniques so as to make testing more efficient and effective.
- **Tools**
    - Phase 1: this phase requires defining the test level (unit test, integration test, system test, etc.) that the company intends to perform. In addition, as there are too many tools that can support different levels of testing there is a need to set selection criteria which considers potential benefits, risks and scripting techniques (data driven/keyword driven).
    - Phase 2: at this phase the company is expected to select and implement appropriate open source test tools as well as arrange and avail the required infrastructure (software, hardware, connectivity, etc.)
    - Phase 3: this phase may require the application of commercial tools that can provide full-fledged features for undertaking comprehensive testing.
- **Test Design & Execution**
    - Phase 1: one of the key activities recommended at this phase is determining the test objectives (i.e. broad categories of things to test) which leads to the creation of a testing matrix reflecting the fundamental elements that needs to be tested to satisfy an objective. In addition, there is a need to gather and organize requirement specifications which must be used as a major input for testing.

- Phase 2: having the clear objective and system requirement set in phase one, the second phase concentrates on conducting basic test execution (preliminary testing, e.g. unit testing and integration testing).This requires selection of test case design techniques, deriving test cases for testing the most common situations and actions and developing test procedure.
- Phase3: at this level the company is expected to undertake compressive test execution involving all levels and types of testing. It also involves review of all test
- process in planning, management and report. The company should also form a team to conduct the validation and verification process and a tester should be a member of the team.

- **Test Documentation**
  - Phase 1: one of the major activities at this phase is identifying major documents including test plan, test design and test case specification, test strategy, test incident reports, test logs, test data, bug report, testing status and test summary reports, weekly status reports, user documents/manuals, risk assessment. In addition, there is a need to systematically organize these documents.
  - Phase 2: At this phase the company should start using systems to manage the above stated documents. One of the cost saving option is using open source document management system
  - Phase 3: the company should deploy a collaborative document authoring and sharing system in order to fully automate the development and sharing of different documents associated with software testing.
  - **III. Standards**:
- Phase 1: the company should at least develop checklist that enables testers to verify the very minimum quality of the software. Checklist can be developed for the most important tasks

- such as GUI testing and platform testing. This checklist should be reviewed and updated to fit the needs of the new projects.
- Phase 2: at this phase the company is expected to derive key indicators from the international standards including TMMi and CMMi with minimalist approach, i.e., by considering the available expertise and resources.
- Phase 3: this phase requires the adoption of international standards including ISO/IEEE/IEC, TMMi and CMMi comprehensively. This should be combined with certifications to be
- Secured by the company in such international standards. Enforcing the application of these standards at the organizational level is also expected from the management.

CHAPTER SIX

VALIDATION

6.1 Introduction

This chapter describes the validation of an integrated software test framework. Firstly, we introduce why we tested the framework and what we expect to receive as the results from the case study. Secondly, we describe the participants and give the reasons why they were selected. We continue and report how we carried out the study and summarize the interviews. Finally, we provide the results of the case study.

6.1 Introduction to an integrated test frame work

The framework provides test managers with the means to evaluate a tool while removing large portion of subjectivity from the process. The test manager can now see which features test software test frame work should support for a company with certain set of characteristics. To confirm the usability of the software test frame evaluation framework, we carried out a case study among three testers. The purpose of the study was to understand whether the frame work fit for use and to find out what should be done to improve the framework’s usability.

6.2 Participant Selection

Three test specialists were contacted and asked to evaluate the frame their companies are using. The limitation of the participants to three persons was due to the limited number of software companies who are currently practicing with better one and have test personnel who have an experience in software testing. Secondly, we wanted to carry out a small proof-of-concept test, not to make a full research on the matter.

The participants were selected from three companies, in order to confirm how the company specific product diagram would be perceived. The study uses three respondents who are working as test engineer’s one person and two were test managers.

6.3 Evaluation Framework Usability Interviews

The study gathered responses to the questionnaire addressing the usability of the test framework evaluation. We used personal approach and performed interviews with the respondents. This provided closer feedback and allowed us to ask additional specifying questions when the answers were vague or superficial. Firstly, we introduced to the participants the purpose of the framework. We explained that the product diagrams are created based on theoretical studies and market research which was later confirmed by performing survey among software companies of Ethiopia companies. The survey results were analyzed, for all test frameworks based on the existing using minimalist approach. The respondents were asked to read the guideline first and then to evaluate their company's software testing as compared to the existing one in their companies by applying the evaluation framework. Additional information was provided when questions regarding the framework were raised.

After the evaluation with the framework, the respondents were requested to respond to the short questionnaire. We were interested in five aspects:

- How easy is the framework to learn?
- How efficient is it for frequent software test framework evaluation?
- How easy is it to remember the activities in each phase of the framework?
- How satisfied are you with the framework?
- How easy it is to understand the benefits of the framework?

The first two interviews revealed that the guideline requires a change. We improved the framework guideline thus making it easier to understand. After that, we proceeded with next respondents. The interviews with the respondents lasted on an average an hour by raising relevant comments by the respondents. Three of the interviews were recorded using phone while two respondents asked us only to make footnotes. The goal of the interviews was to understand whether improvements should be made to the framework and to get feedback on the usability.

6.4 Results of the Case Study

Each of the study participants was asked to give feedback on the framework usability and to rate it on a scale of 1 (lowest) to 5 (highest). We provide the results in Table 6.1 below

Questionnaires	Respondent I	Respondent II	Respondent III
How easy is the framework to learn?	4	4	5
How efficient is it for frequent Test frame work evaluation?	3	4	5
How easy is it to remember the activities in each phase of the framework?	5	4	4
How satisfied are you with the framework?	4	4	3
How easy it is to understand the benefits of the framework?	4	4	4

Table 6.1 Response to validation

The first response about the ease of learning is lower than the others, since improvements were made to the framework guideline based on the interviews. The rest of the survey does not have outstanding differences.

- ***How easy is the framework to learn?***

After applying improvements to the guideline, all respondents considered the framework easy to learn. People understood the workflow how to use the framework. They also implied that there are clear activities are mentioned in each of the phases. Respondents recommended creating the evaluation framework a good approach in understanding the test process

- ***How efficient is it for frequent framework evaluation?***

The respondents understood that for frequent use, testers responded that it would only have use the frame work frequently for sometimes then it is easy to remember. As such, most respondents found that the framework is rather efficient for frequent use.

- ***How easy is it to remember the activities in each phase of the framework?***

Most of the interviewees told that the activities in each of the phase are easy to remember. They said that activities in the phases are logical and quickly followed. One respondent did suggest shortening the activity names, however to keep the framework easy to learn, but due to time constrain we did not make the change.

- ***How satisfied are you with the framework?***

This question turned out to be the hardest to answer. The participants had never used a framework for evaluating a software quality; it was new experience for them. While they did not say they were not satisfied with the framework, they were also reluctant to confirm, that it met their expectations. There was one exception, one of the test managers believed, that evaluating a software test framework should be done by company employee and not based on a framework, since "the employee knows what is required by the company"

- ***How easy it is to understand the benefits of the framework?***

All respondents understood clearly the benefits of the framework mitigation of the subjectivity of evaluation by using an evaluation framework based on structured approach. Similar to the previous question, there was outstanding respondent who strongly believed that the framework would not be beneficial for his company. Despite the outlying result, majority of the interviewees agreed that the benefits are rather easy to understand. Finally, respondents were asked to bring out the best aspect of the framework. Three interviewees told that they got a clear number representing how much the tool met with the company expectations. The other two agreed that the framework is excellent for frequent use and saves time.

6.5 Threats to Validity

We have applied the guidelines (e.g. personal interviews, objective questions, addressing potential risks to validity) suggested by Good *et al.,* (2008) to minimize the threats to the validity of our case study. However there are still few which should paid attention be to when reviewing the results. The first and probably the biggest threat, is the number of participants in the case study. We asked 3 testers to evaluate our framework. The number of the participants was kept low due to the scope of this research. For future work, further analysis should be carried out by including more respondents to the evaluation framework's usability case study. Another aspect which should be mentioned is that the framework validation focused only on the usability and did not address the completeness of the test evaluation framework. To address this risk additional research should be carried out to confirm if all required test features have been included to the evaluation framework. Finally, a threat to the validity of the case study comes from not confirming the correctness of the evaluation framework. We have not investigated if the framework will produce the same results for different respondent groups who evaluate the same test frame work with the test evaluation framework. Our focus was only on the framework usability and thus, the correctness is subject for future work.

6.6 Summary of the Evaluation Framework Testing

The researchers carried out a case study to investigate the usability of the software test evaluation framework. The study involved 3 practitioners and they were asked to evaluate their company's test framework using the test management framework. Each respondent evaluated their companies separately. The result of the evaluation confirms that the framework is easy to learn, efficient for frequent use and fit for purpose. There was one respondent, who was doubtful of the tools suitability for the task, especially the list of test frame requirements. He believed subjective evaluation of the software would meet company's expectations better. However the software test evaluation framework relies on the current selected Ethiopian software companies' expectations, thus, mitigating the subjectivity of test framework evaluation at least in this geographical area. In conclusion, the strongest aspects of the testing frame evaluation framework are that using it frequently is efficient and it gives clear measurable value for the software testing frame work.

CHAPTER SEVEN

CONCLUSIONS AND FUTURE WORK

7.1 Summary of major findings

Software testing is a critical element in the software development life cycle and has the potential to save time and money by identifying problems early and to improve customer satisfaction by delivering a more defect-free product. Software provides a comprehensive tool set for building powerful applications. Without adequate testing, however, there is a greater risk that an application will inadequately deliver what was expected by the business users or that the final product will have problems such that users will eventually abandon it out of frustration. In either case, time and money are lost and the credibility and reputation of both the developers and software tester and company at large is damaged. More formal, rigorous testing will go far to reduce the risk that either of these scenarios occurs.

Software testing is a critical element in the software development life cycle and has the potential to save time and money by identifying problems early and to improve customer satisfaction by delivering a more defect-free product. Software provides a comprehensive tool set for building powerful applications. Without adequate testing, however, there is a greater risk that an application will inadequately deliver what was expected by the business users or that the final product will have problems such that users will eventually abandon it out of frustration. In either case, time and money are lost and the credibility and reputation of both the developers, software tester and the company at large is damaged. More formal, rigorous testing will go far to reduce the risk that either of these scenarios occurs. This study assessed the existing software testing practices, processes and challenges with the aim of proposing an integrated framework that guides software testers in ensuring software quality through appropriate testing. The study addressed three research questions and the results of the study are summarized under these research questions.

The first research question focused on identifying the existing practice and process of software testing in the Ethiopian software companies. According to the findings, software testing process is not given due attention by most software companies in Ethiopia. This is demonstrated by the following existing situations in software companies. Except few, most companies do not have a separate department or team that focus on software testing and they don't perform formal testing process. Most of them assign no or very limited inexperienced staff as tester. Testing is dominantly performed by programmers. Most employees of the companies do not get formal training in software testing and they lack the required up-to-date knowledge and skill. Major test processes like test planning, estimation, design, execution and management are not being properly performed by the dominant number of software companies. Those companies who try to perform software testing also allot very limited time. The testing method is highly dominated by high level review or inspection and manual testing and automated testing is not applied by almost all the companies. Most companies focus only on unit and/or integration testing using white box techniques ignoring other methods and techniques that ensure software quality. The use of appropriate software test tools is also very limited. The use of tools is mainly limited to reporting rather than other testing processes. Almost all companies do not follow any international standard like CMMi and TMMi. The majority of companies do not involve customers in software testing process. Poor documentation; lack of proper communication among project managers, programmers and testers; and demotivation among staff are some of the major problems.

The second research question was mainly concerned with identifying major challenges that Ethiopian software companies are currently facing. Major challenges identified by the study include: lack of test policy/strategy/plan, lack of institutional set-up dedicated for software testing, lack of expertise or skilled staff, lack of adequate budget, lack of proper attention from the owners, lack of appropriate technological infrastructure and lack of proper training and awareness on test processes/methods/techniques/tools.

The solutions recommended by respondents include: recruiting appropriate skilled testers, provision of appropriate training, identifying and purchasing the right test tools, identifying new methods and techniques appropriate for the Ethiopian context, allocating adequate budget and resource, assigning quality assurance personnel, introducing internal audit system, raising organizational commitment, institutionalizing software testing, experience sharing internally and externally, developing or adopting customized standards, enforcement in the application of acceptable standards, provision of adequate infrastructure, proper documentation, creating communication platform and encouraging customer participation.

The third research question focused on exploring the possibility of proposing an integrated framework that can guide software companies to perform effective software testing and ensure software quality in a resource constrained environment. Based on the existing situation of software testing in Ethiopian Software Companies, the prevailing challenges and possible solutions recommended by respondents and other similar studies we proposed a Software Test Improvement Framework. The framework is structured in three major areas of challenge having four sub-categories with proposed activities divided in three phases. Researchers believe that this phased approach enable software companies to introduce formal and effective test processes that ensure software quality given their resource constraints. The originality of the of the framework is signified by the fact that: 1) it is developed based on the empirical findings on the local study context and 2) although the terms are common around software testing literature, the categorizations or classifications as well as prioritization of solutions are typical to the study context. Challenge-based categorization and prioritization of solutions is what we tried to introduce as a new perspective.

7.2. Contribution of the Study

The study will have the following key contributions to both the practice and to the body of knowledge.

- It demonstrates the significance of performing software testing
- It proposes cost saving mechanisms of introducing software testing in the first and second phases of the framework
- It organizes and prioritizes software testing activities in a way a software company can easily implement testing in a more effective way given the existing resource constraint.
- The framework is a contribution by itself for the body of knowledge that can be used as an input for developing theories in the areas of software testing applicable for resource constrained environment

7.3 Recommendation

Based on the findings of the empirical study the following recommendations were made in order to implement phase of software testing activities embedded in the proposed framework.

- The owners of software companies and software project managers should give due attention to software testing process and producing quality software
- Owners and/or managers should create institutional set-up for software testing
- The owners and/or managers should be committed to allocate and avail adequate budget, resources and infrastructure for software testing
- The management should be committed in identifying the key challenges of the company in software testing and addressing those challenges through phased and minimalist approach as it is proposed in the framework.
- The management of software company should focus on better staffing in terms of skilled testers as well as building their capacity with appropriate training
- The management should be committed to enforce the adoption of either customized or international software testing standards at organization level
- The government should institute software quality assurance and monitoring mechanism

7.4 Future Research

Since the present study was restricted only in Addis Ababa in a limited software companies further research should be conducted to investigate the practices and challenges of software companies to enhance the frame work and make it applicable at a national level. In addition, further research should be conducted with the aim of improving the functionality and operationalization of the framework by introducing different categorizations and prioritizations of software testing processes. There is also a need to conduct rigorous validation of the framework by extending the sample size and deploying other validation techniques. One can also select software companies, implement the framework and assess the impact of the framework in the real environment.

Reference

1. Abhijit, A., Sawant, P., Bari ,H. and Chawan, P. M. (201**2):** Software Testing Techniques and Strategies, *International Journal of Engineering Research and Applications*, 2(3), pp.980-986
2. Agarwal, B. B., Tayal, S. P. & Gupta, M. (2010). Software engineering and testing, *Jones*
3. Agarwal, R., Nayak, P., Malarvizhi, M.; Suresh, P., and Modi, N. (2007). Virtual Quality Assurance Facilitation Model .*Global Software Engineering Second IEEE International Conference and Bartlett publishers, LLC.*
4. Alaa, M. El-Halees (2014) Software Usability Evaluation Using Opinion Mining *Journal of Software, 9(2).*
5. *A*nitha, A. (2013) A brief Overview of Software Testing Techniques and Metrics.
6. Arvinder, K., Kaur, B., Suri, S., A. Sharma, *(2007) Software Testing Product Metrics – A survey*. Proceedings of National Conference on Challenges & Opportunities in Information
7. Beizer B., 1990, Software Testing Techniques, International Thomson Computer Press, 2nd edition.
8. Bertolino, A. (2007). Software testing research: achievements, challenges, dreams. *In International Conference on Software Engineering*
9. Black, R. (2000). Investing in Software Testing: The Cost of Software Quality. White paper, RBCS.
10. Boehm, B., Chulani, S., Verner, J., Wong B.,(2007) "Fifth Workshop on Software Quality Software Engineering - *Companion, 29th International Conference*
11. Briand L. and Labiche, Y. (2004). Empirical studies of software testing techniques: challenges, practical strategies and future research, *ACM SIGSOFT Software Engineering Notes, (29) (5), pp. 1-3.*
12. Burnstein, I., Suwanassart, T., Carlson, R. (1996). Developing a testing maturity model for software test process evaluation and improvement, *International Test Conference.*
13. Christer, P. and Nur,Y.(2012) Establishment of Automated Regression Testing at ABB: Industrial 68 Experience Report on 'Avoiding the Pitfalls', *Proceedings of the 19th International Conference on Automated Software Engineering (ASE'04)*
14. Craig, R. and Jaskiel, S. (2002) Systematic Software Testing. Artech House Publishers.
15. Geras, A.M., Smith, M.R., Miller, J., 2004. A survey of software testing practices in Alberta. Canadian Journal of Electrical and Computer Engineering 29 (3), 183–191.

16. Glass,D., and R. L., Collard, R., Bertolino, A., Bach, J., &Kaner, C. (2006). Software testing and industry needs. *IEEE Software, 23,* 55-57.
17. Grover, (2016). Comparison Study of Software Testing Methods and Levels- A Review, 4(8)
18. Hass, A. M. J. (2008). *Guide to advanced software testing.* Norwood, MA: Artech House.
19. Haugset, B., & Hanssen, G. K (2009). Automated Acceptance Testing using FIT, *42nd*
    a. *Hawaii International Conference on System Sciences*, Pp. 1-9
20. Huang, L. and Boehm, B. (2006). How Much is Software Quality .
21. *Huang, L. and Boehm,B.(2007). How Much Software Quality Investment Is Enough: A Value- Based Approach, IEEE Software, Vol. 23(5), pp. 88-95,*
22. Hutcheson and Marnie L. (2003). Software Testing Fundamentals: Methods and Metrics. Wiley Publishing Inc. Indianapolis,
23. International Software Testing Qualifications Board (ISTQB), (2007), Certified Tester Foundation Level Syllabus.
24. Isha, S. (2014)Software Testing Techniques and Strategies Int. Journal of Engineering Research and Applications www, 4(4), pp.99-102.
25. ISO/IEC, ISO/IEC 29119,(2008), Software Testing Standard –Activity Descriptions for Test Process Diagram
26. ISO/IEC/IEEE, (2010) Systems and software engineering Vocabulary, First edition,
27. ISO/IEC/IEEE 29119, (2013) : Software testing Standards
28. ISTQB (2007). International Software Testing Qualifications Board (ISTQB), Certified Tester Foundation Level Syllabus, version 01.05.2007.
29. Jaskiel, and Stefan, P. (2002). Systematic Software Testing.
30. Javed, A., Maqsood, M., Qazi, K. A., & Shah, K. A. (2012). How to Improve Software Quality Assurance in Developing Countries*. Advanced Computing, 3(2), 17*.
31. Javed, A., Muazzam.M., Khurrami A., Khurram, A.(2012): How to improve software quality assurance in developing countries; Advanced Computing: *An International Journal (ACIJ), 3(2)*
32. Jones, C. (2008) Measuring defect potentials and defect removal efficiency. *Crosstalk: The Journal of Defense Software Engineering, 21, 11-13.*
33. Juristo, N., Moreno, A. and Vegas, S.(2004) Reviewing 25 years of testing technique experiment, Empirical Software Engineering, 9(1), pp.7-44
34. Kapur, P.K., (2014) Measuring Software Testing Efficiency Using Two-Way Assessment Technique *QAI Consulting Organization. Emphasizing Software Test Process Improvement,*

35. Karlstrom, D., Runeson, P., &Nordén, S. (2005) A minimal test practice framework for emerging software organizations. *Software Testing Verification and Reliability*, *15*(3), 145-166.
36. Kasurinen, J., Taipale, O., Smolander, K.(2010) : Test Case Selection and Prioritization: Risk based or Design-based?, *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*
37. Katherine, A. V.,& Alagarsamy, K.(2012) Software testing in cloud platform: A survey, *International Journal of Computer Applications*, 46(6)
38. Khan, F. (2012) A Comparative Study of White Box, Black Box and Grey Box Testing Techniques", (IJACSA) International Journal of Advanced Computer Science and Applications, 3(6)
39. Kumar, R. and Chhokra, A. K. (2012).Reduce QA Cost by Improving Productivity& Test Optimization.
40. Leonardo, M., Mauro P., Oliviero, R. and Mauro S.(2014):Software Testing, verification and reliability Published online in Wiley Online Library
41. Loveland, S., Miller, G., Prewitt, R., Shannon, M. (2005) Software testing techniques: Finding the Defects that Matter. Charles River Media, Inc. Hingham, Massachusetts.
42. M. Leonardo, P. Mauro, R. Oliviero, S. Mauro,(2012). Auto Black Test: Automatic Black-Box Testing of Interactive Applications,
43. Maneela T., and Gaurav, D. (2012) A Research Study on importance of Testing and Quality Assurance in Software Development Life Cycle (SDLC) Models. *International Journal of Soft Computing and Engineering (IJSCE), Volume-2(3)*
44. Mansor,Z. and Ndudi,E(2012) Issues, Challenges and Best Practices of Software Testing Activity. Recent Advances on Computer Engineering.
45. Mei, L., Zhang, , Z., Chan, W. K., &Tse, T. H. (2009). Test case prioritization for regression testing of service-oriented business applications.
46. Mockus, A., Nagappan, N., &Dinh-Trong, T. T. (2009) Test coverage and post verification defects: *A multiple case study. In International Conference on Empirical Software Engineering and Measurement.*
47. Myers, G. The Art of Software Testing (2004), *2nd edition, John Wiley & Sons, Inc., New Jersey, USA.*
48. Ng, S.P., Murmane, T., Reed, K., Grant, D. and Chen, T.Y. (2004) A preliminary survey on software testing practices in Australia, Proc. 2004 *(Australian Software Engineering Conference), pp. 116□125.*

49. Nirmala, D. and Maheswari L. T. (2015) Automated Test Framework for Software Quality Assurance. *International Journal of Computer Science and Mobile Computing (IJCSMC), 4 (12), pg.224 – 234.*

50. Olaperi, Y., Sowunmi, M, .Sanjay, Luis, F., Broderick, C., and Ricardo, S. (2016): An empirical evaluation of software quality assurance practices and challenges in a developing country: a comparison of Nigeria and Turkey.

51. Richa, R & Shallu(2013). Performance Evaluation and Comparison of Software Testing Tools. *International Journal of Computer Science & Information Technology*, 3(7), pp: 711-716

52. Schulmeyer, G. (2007). *Handbook of software quality assurance* (4th ed)

53. Sheikhumar,F. and Quadri,S.(2013):Empirical Evaluation of software testing techniques, Needs, Issues and Mitigation .

54. Stuart A. (2011) Regression Testing

55. SWEBOK, (2004). Guide to the Software Engineering Body of Knowledge. IEEE Computer Society, 2004 Craig, Rick D.,

56. Tan, M.T.K. and Hall, W. (2008). Beyond Theoretical and Methodological Pluralism in Interpretive IS Research: The Example of Symbolic Interactions Ethnography, Communications of the Association of Information Systems, 19(1)

57. Tassey G. (2002). The Economic Impacts of Inadequate Infrastructure for Software Testing. *U.S. National Institute of Standards and Technology report, RTI Project*

58. TMMi (2010).Test Maturity Model integration, Version 3.1, TMMi Foundation, Ireland

59. TMMi Foundation (2009).Test Maturity Model integration (TMMi) reference model, Version (2).

60. Venkatesh, V., Brown, S. A., &Bala, H. (2013) Bridging the qualitative-quantitative divide: Guidelines for conducting mixed methods research in information systems.*MIS quarterly*, *37*(1), 21-54.

Questionnaire

ST.MARY'S UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

Dear respondent,

First of all, I would like to thank you in advance for devoting your precious time to fill in the questionnaire. The information that you provide will be used to undertake a study entitled "**AN INTEGRATED SOFTWARE TEST PROCESS: THE CASE OF SELECTED ETHIOPIAN SOFTWARE COMPANIES IN ETHIOPIA"**

The study is part of the requirements for Master of Science in Computer Science. The information you provide will be very confidential, and hence, you are encouraged to freely express your views and concerns. Your data is expected to contribute for the success of the study tremendously. If you have any quires, you may contact me via the address stated below.

Thank you so much for your cooperation in advance

Shimelis Tamiru

E-mail: shimelistamiru4@gmail.com

Mobile:+251-913-289802

Addis Ababa

Please choose the one that you believe is appropriate based on the service experience you have within the software company and put ‘’X’’ mark in the box in front of your choice of preference

I.PROFILE OF RESPONDENTS

1. Sex

☐ Male Female ☐

2. Age

20-30 ☐ 30-40 ☐ above 40 ☐

3. Marital status

☐ Single Married ☐

4. What is your highest formal education attended?

☐ Diploma or below ☐ First Second degree and above ☐

5. Which of the following best describe your current postion?

Project Manager ☐ Software tester ☐ Web designer ☐

Programmer ☐ Requirement Engineer ☐ System Analyst ☐

6. Year of service as a software tester

Less than 1 year ☐ 1-3 year’s ☐ 3-5 years ☐ 5 years above ☐

7. Do you have a software test process in your organization?

Yes ☐ ☐ No

8.. Who is primarily responsible for software testing in your company?

☐ A single, centralized software testing department

☐ Testing staff divided among application groups

☐ No formal software testing staff or department

9. Do you have any formal training in software testing or quality assurance?

☐ Yes ☐ No

10. Please indicate your answer for column (a) and column (b) for the major test processes and sub-activities summarized in the following table.

Test Process and Sub-Activities	(a) Do you practice the following major and sub-activities in software testing process?		(b) (If your answer for column (a) is Yes specify the type of tool and techniques you use
	Yes	No	
Test Planning			▨
			•
			•
			•
Test Estimation			▨
			•
			•
			•
Test Design			▨
			•
			•
			•
Test Execution			▨
			•
			•
			•
Test Management			▨
			•
			•
			•
Test Reporting			▨
			•
			•
			•

11. What percentage of a software tester's time is spent actually on testing an application?

☐ 76%-100%   51%-75 ☐26%-50% ☐   Less than 25% ☐

12. How do you perform software testing in your company?

Activity	Always	Sometimes	Never	N/A	Unknown
Review					
Inspection					
Automated Testing					
Manual Testing					

13. What type of testing methods , techniques and tools being applied by your company (rank the level of utilization for the following Software testing method, techniques and tools)

Methods/ Techniques/ Tools	Very high	high	Moderate	low	Not at all
1.Method/Type					
1.1 Unit Test					
1.2 Integration Test					
1.3 Security Test					
1.4 Performance Test					
1.3 System Test					
1.5 Others ………..					
2.Techniques					
2.1.Black box test					
2.2.White box test					
3.Tools					
3.1.Open source tools					
3.2.Coomercial Tools					
3.3.Locally developed tools					

14. If question number 3.3.3 if you have locally developed software testing please specify its name and purpose.

____

____

15. .Does your companies use automated software testing tools?

☐ Yes ☐ No

16. If your answer is yes, for question number 16 for which software testing process do you apply automated software testing tool?

Test process	**Yes**	**No**	If yes please specify it's name and type (Open , commercial or locally developed
Test Design			
Test Management			
Test Reporting			

17. Do you use any tool regarding the following test methods?

Methods	Yes	No	If yes please specify its name and type (Open, commercial and locally developed
1.Method/Type			
1 Unit Test			
2 Integration Test			
3 System Test			
4.Security Test			
5. Performance Test			
Others ………..			

18. If your answer for question number 19 is poor, what are the major challenges?

____

____

19. Which software testing standard is being applied in your company?

☐ ISO Series ( ) TMMi ☐

☐ CMMi Not at all Exist ☐

20. Are customers encouraged to participate in software testing?

☐ **Yes** ☐ **No**

21. What are the major challenges to the adoption of software testing methods, tools, techniques and test automation? You can choose more than one response.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Methods/Techniques/Tools					
• Lack of expertise					
• Lack of adequate budget					
• Costly to use					
• Difficult to use					
• Time-consuming to use					
• Lack of proper attention					
• I don't think it is useful or cost effective					
• Others					
Test automation					
• Lack of expertise					
• Lack of adequate budget					
• Costly to use					
• Difficult to use					
• Time-consuming to use					
• Lack of proper attention					
• I don't think it is useful or cost effective					
• Others					

22. What criteria does you company uses to exit the testing phase

Criteria	**Always**	**Mostly**	**Sometimes**	**Never**	**N/A**
All planned test activities must have been performed					
Each requirement has been tested at least once					
Delivery time has been reached					
The value specified in the metrics have been reached					
The planned test budget is depleted					
Sufficient defect have been found to have trust in the software					

23. What criteria does your company uses to make a decision as to which level of testing should be implemented

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Availability of Expertise					
Experience of testers					
Availability of adequate budget					
Complexity of the tested system					
Delivery Time					
Other					

24. What do you think in order to solve the current software testing challenges?

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Recruit appropriate skilled testers					
Training existing testers					
Identify and purchase the right testing tools					
Identify new methods and techniques appropriate for our context					
Developing and adopting appropriate testing process model					
Allocate more budget and resource for software test process					
Engage with specialist test provider to benefit from External expertise to maximize the level of importance and benefit					

Interview questions for the research

1. What is the current software testing process in your company?
2. What are the current challenges of the current testing practices in your company?
3. How did you cope up with the challenges faced in your company?
4. Does your organization have Test Policy, Strategy and Plan? If yes please specify the detail.
5. How do you make decision criteria to selecting software method, tools and techniques?
6. Do you make decisions on resources to be allocated in the software testing?
7. What do you suggest in order to improve software testing challenges?

THANK YOU!