Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 3, 303-310 (2021)

www.astesj.com

ASTESJ
ISSN: 2415-6698

Decision Support System for Testing and Evaluating Software in Organizations

Rotimi Adediran Ibitomi, Tefo Gordan Sekgweleo, Tiko Iyamu*

Department of Information technology, Cape Peninsula University of Technology, Cape Town, 8000, South Africa

A R T I C L E   I N F O

Article history:
Received: 13 April, 2021
Accepted: 15 May, 2021
Online: 10 June, 2021

Keywords:
Software Testing
Software Evaluation
Decision Support System

A B S T R A C T

Organizations make use of different tools and methods to test and evaluate software, for quality and appropriateness purposes. However, many organizations are unable to perform multiple testing activities (manual, automation and performance testing), using a single tool. As a result of limitation of some tools, some organizations are hampered in their attempt to test software, which affect productivities. This is a challenge that has for many years, caused severe problems for some organizations, affecting their time to respond to business change. The challenge is associated with the lack of framework to guide complementary use of multiple tools when carrying out software testing. This study employs the case study and the interpretivist approaches. From the findings, a decision support system (DSS) framework was developed, which can be used guide testing and evaluation of software in an organization. Also, the DSS is intended to support and assist software engineers, developers and managers in identifying the factors that influence their decisions from both, technical and non-technical perspectives, when testing software.

1. Introduction

The software process offers the flow of the software and expands the assurance of the software product under production [1]. This first step of test planning is the creation of the test plan which ensures that the testing activities are adhered to and determines precisely what the testing is meant to achieve. The test plan specifies the items to be tested, the level of testing, the sequence of testing, the manner in which the test strategy will be applied to each item, as well as the description of the test environment [2]. With such details the test plan establishes a clear indication to stakeholders, pertaining to the software testing.

Software testing methods are basically the approaches that can be adopted to test and evaluate software within the organization. It points out the direction to follow when conducting software testing. According to Mishra, the two most used types of testing methods are black box (functional) and white box (structural) testing [3]. Black box testing is also referred to as behavioral testing whereby the software is tested without the knowledge of the internal workings of the software [4]. White box testing is where the software tester knows the internal workings of the software [5]. In practice, when performing white box testing, the software tester is granted access to the code, to test the software [6].

At times, it can be surprisingly difficult for people to make decisions, especially when they do not understand the root cause of the problem. Thus, for the decision-makers to overcome limits and constraints encountered, they need *decision support systems* (DSS) to assist them in making difficult decisions for solving complex problems [7]. Decision-making is one of the essential activities of business management and forms a huge component of any process of implementation. Over the years, various DSSs were developed to support decision-makers at all levels in organizations including systems that could support problem structuring, operations, financial management and strategic decision-making, extending to support optimization and simulation [8]. In software testing and evaluation as well, DSSs can assist managers in making the correct decisions.

In assisting managers and software developers including engineers to make decisions and have support for their processes, this empirical study was conducted. The aim was to develop a DSS framework that can be used to test and evaluate software in an organization. Based on this, four research questions were formulated: (1) What are the tools that are used in testing software? (2) What are the methods involved in the testing of the software? (3) What factors influence the testing and evaluating of software in organizations? (4) How can a DSS framework be developed and used to address the challenges occurring during software testing and evaluation in organizations?

*Corresponding Author: Tiko Iyamu, Email: connectvilla@yahoo.com

The remainder of this paper is logically structured as follows: the review of literature related to the study is first presented, followed by the methodology that was applied in achieving the aim of the study. The data analysis and findings follow respectively. The DSS framework and discussion, which are based on the findings are presented before the conclusion.

2. Literature Review

The software automation tools enable software testers to create scripts that can run automatically to test the software. Software automation is valuable to enhance the tester by performing tasks that are tedious, if not impossible for a human, or are more cost effective to automate [9]. Automation testing is when the tester writes scripts for testing the software. Such scripts, running over and over again at no additional cost, are much faster than manual tests, and capable of reducing the time to run repetitive tests from days to hours. Manual testing is considered time consuming, resource intensive and allow some defects to remain uncovered [10]. Therefore, automation testing tools are there to help uncover defects.

Automated testing diminishes the cost of producing software while simultaneously increasing its reliability [11]. Due to the complexity and increasing size of the software, testing efforts are expected to increase. Therefore, automation testing arises as a practical necessity to reduce time and cost [12]. Furthermore, automation testing reduces the amount of manual work, increasing high coverage by executing more test cases and eliminating human errors especially when people tire after multiple repetitions [13]. Automation testing will also likely rectify some of the other problems, but it certainly is not a panacea for solving all problems [14]. In fact, it is always best to possess the skills for utilizing automation tools because without these skills, it is pointless possessing such tools.

The test process describes the test analysis and design as the activity of designing the test cases using the techniques selected during planning [15]. This can be successfully achieved if the software tester understands the user requirements. However, if the software tester fails to understand the user requirements and architecture software under test, it would not be possible to create test cases which will reveal more errors in a shorter amount of time [16]. A test case outlines the steps required to test any functionality of the software and contains the expected and actual results [1]. Such a comparison means the software tester can readily determine whether the software under test satisfies the requirements or functions properly. These test cases could be captured in a spreadsheet or a testing tool if the organization has one.

DSSs are designed to assist individuals or groups with decision-making and solving problems. DSS is described as “interactive computer-based systems that help people use computer communications, data, documents, knowledge and models, to solve problems and make decisions” [17]. Also, the DSS is considered to replace human decision-making as they assist humans to make informed decisions regarding problems they are facing, thereby enhancing the decision-making process. DSSs are not designed to automate decisions but to fairly support decision making because they are flexible enough to react to changing requirements [18]. Typically, most DSSs have three main components: a model system, a data system and user interface [19].

Each component fulfils its particular activity within the DSS. The data relating to the problem is stored in the knowledge base, the model generates decisions based on the content of the knowledge, and the user interface allows users to build models and attain decision support through the adjustment of input parameters [20]. DSSs are available to managers in support of decision-making processes for solving complex issues [21]. Therefore, decision makers can utilize these tools to compile useful information from documents, raw data, and personal knowledge to make decisions in solving problems. Some DSSs give structured information directly to managers and store knowledge which is availed to managers anytime deemed necessary [22].

3. Methodology

The qualitative approach enables researchers to capture the thoughts and feelings of the research participants, leading to an understanding of the meaning that people ascribe to their experiences [23]. As it provides rich descriptive accounts of the phenomenon under investigation [24], it is concerned with addressing the social aspects of the world and seeks to find answers regarding people's behavior, opinions, cultures and differences between social groups [25]. Qualitative research is effective in understanding and explaining complicated situations by obtaining daily knowledge to create theories [26].

A case study as “a contemporary phenomenon within its real-life context, especially when the boundaries between a phenomenon and context are not clear and the researcher has little control over the phenomenon and context” [27]. A high-quality case study puts emphasis on rigor, validity, and reliability [28]. In addition, a case study is suitable in an environment where there are large numbers of variables in a small number of applied units of analysis where the context is of great importance [29]. Three South African based organizations were used as cases in the study. Some details about the organizations are provided below. Twelve private and public organizations were identified, of which only seven had software testing units. From the seven, only three were prepared to participate in the study.

3.1. Mootledi Logistics

Mootledi Logistics is in Kempton Park, Gauteng province. It is a car rental organization that renders services locally and internationally. It relies on software it purchases from vendors as well as the software it develops in-house to serve its customers and perform day-to-day duties. However, such software requires testing and evaluating prior to its deployment into production to ensure that business continued as usual without interruptions. Participants who were involved for the purpose of this study were the employees of Mootledi Logistics. Initial interviews and follow-up interviews were conducted at the organization’s premises in Kempton Park.

3.2. Mmuso Technologies

Mmuso Technologies is in Centurion, also in Gauteng province, with branches in various provinces around South Africa. Mmuso Technologies is responsible for providing government departments with software solutions to fulfil their duties. Once the software is developed, tested, and evaluated it is then accepted by the government department that requested it, and adopted to perform day-to-day activities of that particular department. The software is used to render necessary services to the citizens of the Republic of South Africa. Participants who were involved for the purpose of this study are the employees of Mmuso Technologies.

3.3. Bokamoso Solutions

Bokamoso Solutions, situated in Illovo, Gauteng, is an organization which invests in research to ensure its convergence and relevance to the industry, thereby providing solutions that add value to its market base. Bokamoso Solutions employees are based at the client site to offer their specialized services. Those services include software development, infrastructure management, software testing and consulting.

The semi-structured interview technique was used to collect data from participants from the three organizations. As shown in Table 1, a total of thirty-nine people (technical x 28, business x 11) participated in the study. The interview was concluded at the point of saturation, which means that new information was not forthcoming.

Table 1: Participants in data collection

Case	Technical (IT)	Non-Technical (Bus.)
Mootledi Logistics	Project Managers x1	Business Managers x1
	Software Developers x2	Business Analysts x1
	Software Testers x2	Business Users x1
	Systems Analysts x1	
	Support Specialist x1	
	IT Managers x1	
Mmuso Technologies	Project Managers x1	Business Managers x1
	Software Developers x2	Business Analysts x1
	Software Testers x3	Business Users x1
	Systems Analysts x1	
	Support Specialist x1	
	IT Managers x1	
Bokamoso Solutions	Project Managers x2	Business Managers x2
	Software Developers x3	Business Analysts x1
	Software Testers x2	Business Users x2
	Systems Analysts x2	
	Support Specialist x1	
	IT Managers x1	
Total	28	11

In this study, data was analyzed using the hermeneutic technique from the interpretive approach. Hermeneutic was selected as it allows for a phenomenon to be studied subjectively. Hermeneutic focuses on reality as a human contrast, which can only be understood subjectively [30].

In conducting the analysis, the hermeneutic method of the interpretivist paradigm was applied. The Hermeneutic focuses on how humans construct meanings socially [31]. The method was applied to the following:

i. the meanings which individuals give or associate to events and artefacts.
ii. how those meanings manifest while providing health services.
iii. how the meanings were used to interact with the rules and regulations within the environment.

4. Data Analysis

The data from the three cases were combined. The analysis was conducted based on the research questions:

4.1. What are the tools that are used in testing software?

Software testing can be performed manually or with software testing tools, tools which are either free, open source or proprietary software. Open-source tools are often provided freely by those who developed them, free to be downloaded from the Internet. Proprietary tools, on the other hand, are commercialized, which means that they require licensing rights per user. The tools can be cost prohibitive, making affordability difficult for some organizations, from a purchase point of view. As a result, some organizations opt for using a Microsoft spreadsheet to capture test requirements, test cases and log defects. Other organizations though, can afford to invest larger amounts of money in software testing. This investment is used to set up independent software testing teams and purchase software testing tools that enable them to more readily perform various types of testing.

However, there is no single software testing tool that allows an organization to perform multiple testing activities such as manual, automation and performance testing. As a result of tool limitations, the organization fails to achieve its objectives in conducting end-to-end testing. Software testing needs to cover all requirements, both functional and non-functional. Therefore, the software testing team cannot claim to have produced quality software if they have only covered functional requirements or non-functional requirements as opposed to both. Consequently, organizations need to acquire these tools to cover the entire spectrum of software testing. For organizations it is costly, but for the companies producing these tools, it is profitable. The main reason for separating these tools is for the suppliers to make profits.

Having explored all three cases, one organization opted purely for open-source tools. However, they could not use Selenium effectively to perform automation testing because they relied on self-training. There was therefore not enough time for them to learn because they were also involved in the development, testing and evaluation of the software. As there was no independent software testing team, there was a lack of skill regarding software testing in this organization. As a result, poor quality software was produced.

The second organization purchased IBM rationale tools which enabled them to capture test requirements, test cases and log defects. Due to low budget, however, they adopted JMeter (open source) for performance testing. At least they had an independent software testing team, a manual testing tool and a performance testing tool. They had the advantage of producing quality software because they were able to cover both functional and non-functional testing. The only disadvantage was that they did not have an automation tool to accelerate the testing and evaluation of the software. Therefore, they required too many testers to perform manual and functional testing.

4.2. What are the methods involved in the testing of software?

Software testing methods are the approaches that can be adopted by the software testers to test and evaluate software within the organization. Thus, software testers are trained and equipped with the necessary testing knowledge to assist with testing the software. These testing methods include black box and white box testing which have been discussed extensively in the introduction. Experienced software testers know which testing methods to follow and when. For example, black box testing is performed when the software tester does not know the internal workings of the software. Those who know the internal workings of the software and have programming knowledge perform white box testing. Software testers with limited knowledge of the software conduct grey box testing.

Not anyone is qualified to be a software tester. Software testing is a career field just like software development, project management and business analysis. There are international software testing standards approved by ISO which need to be adopted to produce quality software. Software testing is process-oriented; therefore, software testers must follow testing processes and frameworks to deliver quality software. It is an intense process which requires software testers to carefully follow a set of steps, instructions, guidelines and policies to produce quality software. The lack of framework, standards and procedures within the organization compromises the software quality. It is like picking up someone from the street who does not have a clue about software testing and simply instructing them to test the software.

4.3. What factors influence the testing and evaluating of software in organizations?

Software is a product and therefore, like every product released to the public or organization, needs to undergo testing. The software testing team needs to verify and validate whether the software behaves as expected. They need to test and evaluate both the functional and non-functional requirements of the software. Performing functional testing enables software testers to detect defects which could be fixed while the software is still undergoing testing. Detecting defects in production is risky because it hinders business, it taints the image of the organization and impacts negative customer reaction. Quality software that sustains and enables organizations to be competitive must be delivered to businesses. However, should the software work as expected, it does not necessarily mean it will automatically function, without testing the non-functional requirements. Performance testing is necessary to determine the response and stability of the software under countless workloads. It measures the quality attributes of the software such as scalability, reliability, and resource usage. Some public and private companies in Gauteng province encountered performance challenges whereby their software could not handle the load of users accessing their software. Therefore, it is vital for businesses to cover all the requirements when testing the software.

4.4. How can a DSS framework be developed and used to address the challenges occurring during software testing and evaluation in organizations?

The analysis and interpretation of the data indicate that if the challenges that occurs during software testing and evaluation are not addressed, they will continue to impact the quality of the software negatively. The consequences of not addressing these challenges will result in the software project not being implemented or perhaps worse, being implemented with defects. As a result, the organization's challenges will negatively impact business and customers.

From the analysis, certain factors were found to clearly influence the testing and evaluation of software. Based on the interpretation of these factors, a framework was developed. The DSS framework for testing and evaluating software was designed to assist with addressing the challenges occurring during the testing and evaluation of software in organizations.

4.5. How the objectives of the study were achieved

From the analysis of the data, interpretation was carried out. This was to further gain a deeper understanding of the factors that influence decisions that are made in software testing. The factors were used to develop a DSS. The interpretation is grouped into four categories to ease understanding as follows:

- Examine and understand the tools

This was primarily to examine and gain an understanding of how manual, automation and performance testing was carried out for software testing. There was an evident lack of management buy-in within the organization. It was evident that management was not willing to invest money in software testing. Firstly, there was no independent software testing team to test and evaluate the software in order to produce the quality of software within the organization. Employees who specialized in other fields such as business analysis and software development were tasked to do the software testing. As software testing is a specialized skill, the organization needed to utilize trained software testers to perform software testing. Secondly, free open-source software testing tools were adopted: instead of purchasing proprietary tools, the organization settled for free open-source software testing tools. Employees researched these tools, trained themselves on these tools and adopted those tools. However, they could not fully utilize the tools but used it only to perform automation testing. As a result, proper quality software could not be achieved, but this research objective.

- Explore and understand methods

The approaches adopted for testing software, white or black box was examined. The following factors indicated that quality was not taken seriously: process-oriented, lack of framework, lack of standards and procedures, software evaluation. Software testing is process-oriented. It is an intense process which requires software testers to follow a particular set of steps, instructions, guidelines and policies to produce quality software. Therefore, if the organization does not have a dedicated software testing team trained to perform software testing, they would not know how to test, what to test and when to test what. As software testing is procedural, software testers follow a particular sequence to execute their testing activities. Also, if there is no testing framework, no testing standards and no procedures, employees would not know how to test and evaluate the software. As a result, quality software cannot be delivered. This objective was also achieved.

- Examine the factors

We wanted to examine the factors that trigger software testing, which at the same time influences the testing and evaluating of software in an organization. Organizations rely on software for competitiveness and sustainability. Therefore, organizations must continue to develop software while also enhancing existing software. It is evident that this creates a need and influences the software testing and evaluation in organizations. However, in order for the software testing team to be able to test, they require documentation such as business requirement specifications and technical design specifications. They must follow testing standards and procedures when conducting software testing. Both functional and non-functional requirements must be integrated to achieve quality software. Various teams interact and work together to deliver quality software. Therefore, this objective was achieved.

- Based on the findings from the aim as stated above, a DSS framework is created

The aim of this DSS framework is to address the challenges that occur during software testing and evaluation in organizations. The DSS framework for testing and evaluating software in organizations was achieved based on the findings from the three selected organizations. Therefore, any organization either private, small to medium, or public may adopt this framework in testing and evaluating software. This framework, when followed, will guide the organization in delivering quality software.

5. DSS Framework for Testing and Evaluating Software

From the findings and interpretation thereof, seven factors were found to have a critical influence on the testing and evaluation of software within an organization: requirements, methodology, filtering, repository, governance, assessment, and institutionalization. Figure 1 depicts these factors and how they relate to each other. Each of the factors consist of phases (P1 . . .P^{+n}) in the sequential order of the activities. Some of the activities can be implemented in parallel. For example, policy, standard and principles within governance can be carried out concurrently. To understand the framework, the discussion should be read in conjunction with the factors mentioned in Figure 1 below.

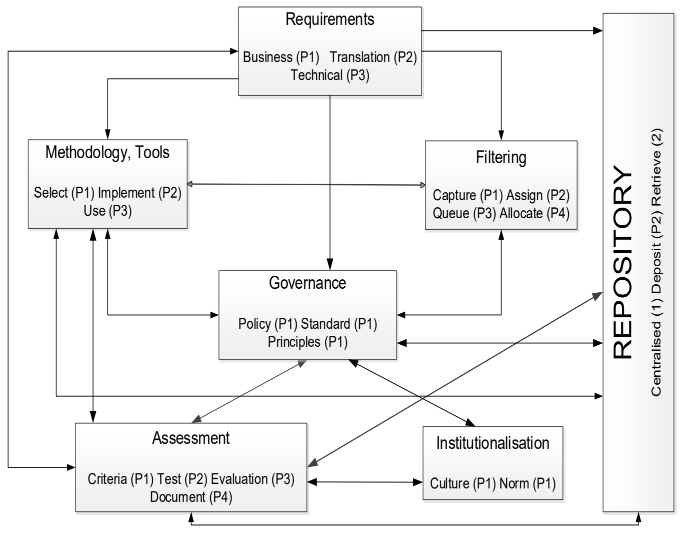
Requirements
Business (P1) Translation (P2) Technical (P3)
Methodology, Tools
Select (P1) Implement (P2) Use (P3)
Filtering
Capture (P1) Assign (P2) Queue (P3) Allocate (P4)
Governance
Policy (P1) Standard (P1) Principles (P1)
Assessment
Criteria (P1) Test (P2) Evaluation (P3) Document (P4)
Institutionalisation
Culture (P1) Norm (P1)
REPOSITORY
Centralised (1) Deposit (P2) Retrieve (2)

Figure 1: DSS framework for testing and evaluating software

5.1. Requirements

In an organization, software testing and evaluation requirements could be either functional or non-functional, in both business and technical units, respectively. The functional requirements basically describe what the software should do. Some of the typical functional requirements include factors such as business rules, authentication, external interfaces, reporting and administrative functions. The non-functional requirements describe how the software should technically behave within the environment. Non-functional requirements cover all the remaining requirements not covered by the functional requirements. The non-functional requirements specify the criteria used for the assessment of software in an organization. For example, the software should be able to coexist with other software in the environment. Also, the non-functional requirements elaborate a performance characteristic of the software. Some characteristics of non-functional requirements include response times, throughput and utilization of the software.

Manual software testers and automation testers extract the functional requirements from the business requirements they receive from the business analysis team. These functional requirements enable the creation of test requirements and test cases which are then captured in the software testing tools adopted by the organization. A test case outlines the steps required to test any functionality of the software and contains the expected and actual results [1]. Test cases are basically scenarios that have been identified from the requirement specification. These test cases can be automated by automation testers to assist the manual testers with regression testing. Therefore, organizations can purchase software testing tools or utilize open-source tools, depending on the investment the organization is willing to make in software testing.

Performance testers make use of non-functional requirements to test the performance of the software, monitoring the software's continuous load on the network. For example, potential leaks can be detected in memory utilization along with analysis of performance degradation and how the software copes under strenuous use. Performance testing is also performed through software testing tools adopted by the organization. Testing concurrent authentication, for example, means that a hall would be filled with a number of users that need to log in concurrently on the software tested. As it would be too expensive to get, for example, 1000 users in the same room to log into the particular software at the same time, a database nowadays locks input into the database for creating records per user. Therefore, it queues the input and creates individual records in the database.

Both functional and non-functional requirements get stored in the software testing tool used by the organization. Some organizations that do not have these tools, create test cases on spreadsheets and store them in a repository such as Microsoft SharePoint, Hyperaware, intranet or a shared drive. This storage enables any other team that might need that information to easily retrieve it.

5.2. Methodology

Software testing methodologies are the different ways of ensuring that the software under test is fully tested. Methodologies and tools are selected and implemented based on organizational requirements. Software testing methodologies encompass functional and non-functional testing to validate the software under test. The testing methods include unit testing, integration testing, system testing and performance testing. As software increases in complexity and is enmeshed with the large number of different platforms and devices that need to be tested, it is more important than ever to have robust testing methodologies, ensuring that software being developed, has been carefully tested. This is to ensure that the software meets its specified requirements and can successfully operate in all anticipated environments with the required usability and security.

The software testing methodology has a definite test objective, test strategy and deliverables. Irrespective of which software development methodology (traditional or agile) has been adopted within the organization, the testing methodologies stated above can be applied in the testing and evaluation of software depending on the scope of the project. However, to successfully apply some of these methodologies, software testers require software testing tools. For example, performance testing is performed to determine the performance of the software or network in terms of response, speed and stability, under a particular workload.

Both human and non-human actors are dependent on each other to test and evaluate software. This is to ensure that quality software is delivered to those who requested it. Software testers, including manual, automation, and performance, require software testing tools to perform various types of software testing, as stated above. These tools could be proprietary or open source. LoadRunner is a tool that can be used in determining the performance and outcome of the software under load [32]. Software testing tools enable software testers to successfully perform their duties.

5.3. Filtering

Filtering in software testing and evaluation is a process of removing unwanted functionality or defects from the software, a process guided by requirements as illustrated in the framework (Figure 1). The main activities of the process are as follows: (1) capture the software into the systems; (2) thereafter the software is assigned to a domain; (3) this is put in a queue; (4) and then it is allocated to personnel for testing and evaluation. The fact is that human beings are prone to making mistakes. For example, the business analysts can incorrectly state the business rule in the requirement specification or specify a requirement ambiguously. If the mistake is not detected by the software developer, the business rule will be built into the code and the software will not behave as expected. When software testers are testing the software, the incorrect business rule would be detected as a defect because the software will not be functioning as expected. Therefore, detecting such defects filters unwanted functionality from the software. Regarding ambiguous requirements, the software tester will not be in a position to create some test cases due to unclear requirements. As a result, clarity regarding those requirements would be required from the business analyst. This, then, is the filtering process in terms of requirements.

When compiling test plans, the test managers can identify risks from the requirement specifications. Also, during execution of test cases, risks could be identified which might impact detrimentally on the quality of the software. These risks must be mitigated to produce quality software. Defects detected and logged during the testing and evaluation of software form part of the filtering process and it is the responsibility of the software testing team to filter all unwanted materials from the software under test in their efforts to deliver quality software.

5.4. Repository

A *repository* generally refers to a central place where information gets stored, accessed and maintained. The repository is defined by the organizational requirements. Activities of governance and assessment including the methodology and tools that are applied for testing and evaluation are stored in the repository, primarily to enable and support the ease of access to stock of organizational knowledge, which fosters quality testing and evaluation of software. Additionally, a repository enables control of organizational stock. Thus, those who wish to access the information that is stored in the repository must apply for access to retrieve whatever information they seek. All materials or information stored in the repository must be secured at all times to protect organizational information against attack and leakages.

There are other documents prepared by the software testing team which include statements of work, test plans and closure reports which must also be stored in the repository for audit purposes. Auditors require these documents during auditing to validate how the organization tested and evaluated their software. This information assists them in compiling their audit reports for the organization they are auditing. Also, the information that resides in the software testing tools such as test cases, execution of those test cases and defects logged and fixed for particular software, assists auditors in validating whether or not the software was tested and evaluated according to software testing standards. All these audit findings aid the organization in fixing their mistakes and improving how they test and evaluate software.

5.5. Governance

Governance plays a pivotal role in the process of evaluating the quality of the software. Governance includes policy, standards and principles which can be applied to the process of software testing and evaluation concurrently. Governance involves the definition of organizational test processes, test documentation and the derivation to testing techniques. The software testing process affords the organization with governance on ways to implement the adopted testing policy, standards and principles that aid the stakeholders to deliver quality software. Test governance enforces compliance to the organization’s testing process. Governance also ensures that the testing processes are continually improved to ensure the constant, uninterrupted delivery of quality software.

Key documentation such as the test policy and the organizational test strategy requires management support as these documents form part of the organizational test processes. The testing processes can be broken into three parts: organizational and dynamic test processes, and test management processes [33]. All three test processes have key documentation that goes along with them for a successful testing organization. The test policy, for example, defines the overall principles that guide testing in the organization. This document is the primary testing document informing the entire testing organization of why software testing is performed. All other testing documentation and processes are based on the test policy. It should include the test policy statement, policy principles and testing standards, as it is the foundation provided for the testing processes [33].

There are various standards available in the testing industry, IEEE's is common because it has a series of five standards [33]. The purpose of the standards is to provide generally acceptable methods of testing across the entire testing industry. Organizations can choose to comply fully with these standards or opt for partial compliance. The standards are not only limited to international standards, the organization can still produce and implement internal testing standards that can best fit the local context of their organization.

The principles of software testing, regarded as the beliefs of the testing organization, remind the software testers within the organization of the reason they test and how they test in the first place. The principles form the foundation of the software testing organization and are clearly articulated in the testing policy. These generally accepted testing principles are formulated by testing industry bodies like the International Software Testing Qualifications Board (ISTQB). These principles form part of the early training and development curriculum in software testing.

5.6. Assessment

Assessment promotes quality of software in an environment. Thus, many organizations find value in benchmarking their progress to improve their processes through assessment. Organizations that develop or enhance existing software have test processes in place to test and evaluate their software. The software testing processes begin with test planning, designing of test cases, preparing for execution and evaluating status till the test closure. During the test planning, the scope and risks, test approaches and testing objectives are identified, enabling the software testing team to identify how much testing needs to happen and what possible risks might be encountered.

The next stage is the analysis and design where software testers identify test conditions, evaluate the testability of requirements and the test environment is set up. During the test implementation, test cases are prioritized, and test data is created for those test cases and then executed. Once the execution is complete, the software testing team reports on the outcome of the testing and the closure report is compiled. Organizations that test and evaluate software have their own testing processes in place.

Therefore, organizations must assess their software testing process to improve the way in which they conduct software testing. It provides the opportunity for the organization to know itself and its competition better. As a result, the organization can strive to produce quality software that will enable it to out-compete its rivals. Through requirements, organizations are able to: identify testing objectives, determine the scope of testing, determine which testing approaches to employ and determine risks that might be incurred during testing. All these activities are documented and stored in the repository for future reference.

5.7. Institutionalization

Institutionalization is a state of stability that is required in software to guarantee quality. It is ways, such as continuous assessment and adherence to governance, in which the software team performs their daily activities for testing and evaluation of software. Iyamu defines *institutionalization* as the process where practices are assimilated into the norm: the ways in which project stakeholders perform their testing and evaluation of software, finally becomes the organizational norm [34]. Those norms should match the internationally agreed set of standards for software testing applicable within any organization. By implementing these standards, the organization will be adopting the only internationally recognized and agreed standards for software testing, giving the organization a high-quality approach to testing software.

Such norms become the software testing culture that is adopted by software testers within the organization. The culture of software quality must be practiced in all parts of the organization because quality is essential for success. Software teams involve various stakeholders such as project managers, business analysts, software developers, software testers, designers, product owners and executive. All these stakeholders play a role in the quality of the final software. Due to this, they need to align their work and practices with agreed international standards, best practices and the test maturity levels in order to deliver quality software. Quality software enables the organization to compete locally and globally with other organizations.

Finally, institutionalization and norms manifest into the organization culture. Organizational culture is a combined means of regulating the behavior of employees within an organization which diffuses all activities as catalysts for the development and growth of the organization [35]. Should the foundation of this organizational culture hinge upon incorrect norms, then the organization would not be in a position to produce quality software. Thus, organizations need to align themselves with the best software testing practices, ISO testing processes as well as testing maturity models. In so doing, the best organizational culture will be practiced by employees. The organizational culture must be documented and stored in the repository so that new employees joining the organization can learn how quality software is produced within the organization. Moreover, existing employees can remind themselves of certain practices they may have forgotten.

6. Conclusion

Organizations are still facing the challenge of ensuring that software projects are tested and evaluated successfully so as not to disrupt business. However, organizations tend to invest in technology and place less emphasis on equipping software testers with software testing knowledge and skills. Hence the same challenges continue to repeat themselves over the years. The contribution of this study is mainly on empirical evidence, giving confidence to employers and employees in adjusting and managing the processes and activities in the testing and evaluation of software in their various organizations. The framework outlines the organizational activities, technologies and governance. The organizational activities include culture, people and operations. Technologies cover the dimensions of the systems, innovation and adoption. Governance includes transformation, awareness and collaboration. Therefore, IT decisions needed to be made to achieve the organizational strategy. This framework can be adopted by any organization that intends to use technology to implement its organizational strategy.

References

[1] I. Hooda, R.S. Chhillar, "Software test process, testing types and techniques," International Journal of Computer Applications, **111**(13), 10-14, 2015, doi:10.5120/19597-1433.

[2] S. Agarwal, P. Sharma, K. Nikhil, 2012. "A review of the software testing process in SDLC," International Journal of Electronics Communication and Computer Engineering, **3**(1), 18-21, 2012.

[3] D. Mishra, S. Ostrovska, T. Hacaloglu, "Exploring and expanding students' success in software testing," Information Technology & People, **30**(4), 927-945, 2017, doi/10.1108/ITP-06-2016-0129.

[4.] T. Hussain, S. Singh, "A comparative study of software testing techniques viz. white box testing black box testing and grey box testing," International Journal Peer Reviewed Refereed, **2**(5), 1-8, 2015.

[5] M.A. Jamil, M. Arif, N.S.A. Abubakar, A. Ahmad, "Software testing techniques: A literature review," 6th International Conference on Information and Communication Technology for The Muslim World, 177-182, 2016, doi:10.1109/ICT4M.2016.045.

[6] S. Nidhra, J. Dondeti, "Black box and white box techniques - A literature review," International Journal of Embedded Systems and Applications, ***2***(2), 29-50, 2012, doi:10.5121/ijesa.2012.2204.

[7] F.G. Filip, C.B. Zamfirescu, C. Ciurea, Computer-Supported Collaborative Decision-Making. Springer International Publishing, 2017.

[8] S. Liu, A. Duffy, R.I. Whitfield, I.M. Boyle, "Integration of decision support systems to improve decision support performance," Knowledge and Information Systems, **22**(3),261-286, 2010, doi.org/10.1007/s10115-009-0192-4.

[9] D. Hoffman, "Test Automation Architectures: Planning for Test Automation," International Quality Week, 37-45, 1999.

[10] M. Kaur, R. Kumari, "Comparative study of automated testing tools: TestComplete and QuickTest Pro," International Journal of Computer Applications, **24**(1):1-7, 2011, doi:10.1.1.259.4366.

[11] D. Shao, S. Khurshid, D.E. Perry, A Case for white-box testing using declarative specifications poster abstract. *Testing: Academic and Industrial Conference Practice and Research Techniques*, IEEE, 137-137), 2007, doi:10.1109/TAIC.PART.2007.36.

[12] G. Mandi, P. Kumar, "Reducing the size of test suite using a variant of non-dominated sorting genetic algorithm II,". International Journal of Advance Research in Computer Science and Management Studies, **1**(6), 66-75, 2013.

[13] A. Nawaz, K.M. Malik, Software Testing Process in Agile Development. Department of Computer Science School of Engineering, Blekinge Institute of Technology, 2008.

[14] J. Tretmans, "Testing concurrent systems: a formal approach," International Conference on Concurrency Theory, 46-66, 1999, doi.org/10.1007/3-540-48320-9_6.

[15] J. Kasurinen, "Software Organizations and Test Process Development," Advances in Computers, **85**,1-63, 2012, doi.org/10.1016/B978-0-12-396526-4.00001-1

[16] S.M.K. Quadri, S.U. Farooq, "Software testing – goals, principles, and limitations," International Journal of Computer Applications, **6**(9), 7-10, 2010, doi:10.5120/1343-1448.

[17] P. Hertz, S. Cavalieri, G.R. Finke, A. Duchi, P. Schönsleben, "A simulation-based decision support system for industrial field service network planning;" Simulation, **90**(1), 69-84, 2014, doi:10.1177/0037549713512685.

[18] P. Hertz, S. Cavalieri, G.R. Finke, A. Duchi, P. Schönsleben, "A simulation-based decision support system for industrial field service network planning," Simulation: Transactions of the Society for Modeling and Simulation International, 1-16, 2013, doi: 10.1177/0037549713512685

[19] B.A. Engel, J.Y. Choi, J. Harbor, S. Pandey, "Web-based DSS for hydrologic impact evaluation of small watershed land use changes," Computers and Electronics in Agriculture, **39**, 241-249, 2003, doi.org/10.1016/S0168-1699(03)00078-4

[20] S. Hosio, J. Goncalves, T. Anagnostopoulos, V. Kostakos, "Leveraging wisdom of the crowd for decision support," In Proceedings of the 30th International BCS Human Computer Interaction Conference: Fusion, 1-12, 2016, doi:10.14236/ewic/HCI2016.38.

[21] A. Athanasiadis, Z.S. Andreopoulou, "DSS applications in forest policy and management: Analysis of current trends," In International Conference on Information and Communication Technologies, 549-557, 2011, doi:10.13140/RG.2.1.4117.6404.

[22] A. Athanasiadis, Z. Andreopoulou, "A DSS for the identification of forest land types by the Greek Forest Service," International Journal of Sustainable Agricultural Management and Informatics, **1**(1), 76-88, 2015, doi: 10.1504/IJSAMI.2015.069053.

[23] J. Sutton, Z. Austin, "Qualitative research: Data collection, analysis, and management," The Canadian Journal of Hospital Pharmacy, **68**(3), 226–231, 2015, doi:10.4212/cjhp.v68i3.1456

[24] O. Gelo, D. Braakmann, G. Benetka, "Quantitative and Qualitative Research: Beyond the Debate," Integrative Psychological and Behavioral Science, **42**, 266–290, 2008, doi: 10.1007/s12124-008-9078-3

[25] J. Barker, P. Linsley, R. Kane, R. 2016. Evidence-based Practice for Nurses and Healthcare Professionals. 3rd ed. Sage Publications Inc, 2016.

[26] M. Petrescu, B. Lauer, "Qualitative Marketing Research: The State of Journal Publications," The Qualitative Report, **22**(9), 2248-2287, 2017, doi:10.46743/2160-3715/2017.2481.

[27] B. Yazan, "Three Approaches to Case Study Methods in Education: Yin, Merriam, and Stake," The Qualitative Report, **20**(2), 134-152, 2015, doi:10.46743/2160-3715/2015.2102.

[28] R.K. Yin, Case Study Research Design and Methods. 5th ed. Thousand Oaks, CA: Sage, 2014.

[29] S. Henderson, Research Methodology. International Journal of Sales, Retailing and Marketing, **4**(9),1-97, 2016,

[30] J. H. Kroeze, "Postmodernism, Interpretivism, and Formal Ontologies". Proceedings in: Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems, British Cataloguing, 43-62, 2012, doi:10.4018/978-1-4666-0179-6.ch003.

[31] S. Robinson, R. Kerr, "Reflexive conversations: Constructing hermeneutic designs for qualitative management research," British Journal of Management, **26**(4), 777-790, 2015, doi.org/10.1111/1467-8551.12118.

[32] S. Sharmila, E. Ramadevi, "Analysis of performance testing on web applications," International Journal of Advanced Research in Computer and Communication Engineering, **3**(3),5258-5260, 2014.

[33] SO/IEC/IEEE International Standard. 2013. Software and systems engineering -- Software testing --Part 3: Test documentation. ISO/IEC/IEEE 29119-1:2013(E), 1-64. IEEE, 2013.

[34] T. Iyamu, "Institutionalisation of the enterprise architecture: The actor-network perspective," International Journal of Actor-Network Theory and Technological Innovation, **3**(1), 27-38, 201, doi:10.4018/978-1-4666-2166-4.ch012.

[35] G. Gavric, G. Sormaz, D. Ilic, "The Impact of Organizational Culture on the Ultimate Performance of a Company," International Review, **3**(4):25-30, Medimond Publishing Company, 2016,