JENRS

Received: 30 October, 2024, Revised: 23 November, 2024, Accepted: 23 November, 2024, Online: 19 December, 2024

DOI: https://doi.org/10.55708/js0312001

Exploring Challenges in Software Testing: A Structuration Theory Perspective

Tefo Gordon Sekgweleo [*1], **Phathutshedzo Makovhololo** [2]

[1] Eskom, Department, Research, Testing & Development, Johannesburg, 2095, South Africa

[2] Cape Peninsula University of Technology, Informatics, University, Cape Town, 8000, South Africa

E-mails: Ts330ci@gmail.com / phathuts@gmail.com

*Corresponding author: Dr Tefo Gordon Sekgweleo, Lower Germiston Rd, Rosherville, Johannesburg, 2095, 082 533 3484 & Ts330ci@gmail.com

ABSTRACT: Developing software is a huge job, which is why digital product teams rely on the software development life cycle (SDLC). SDLC is a critical framework for digital product teams, and software testing is its most vital component. Testing evaluates software components to identify properties of interest, detect defects, and ensure alignment with requirements. If not optimized, testing can be costly, and its omission or inadequate execution can lead to software failures, compromising business operations and reputation. This study explores the challenges of software testing, adopting an interpretivist approach with semi-structured data collection and analysis guided by Structuration theory's duality of structure. The key findings are: (1) Software testing is crucial for delivering quality products and services, ensuring that software meets client requirements and is free from defects. (2) Effective communication and collaboration among agents, including software testers, developers, and project managers, are vital for successful software testing outcomes. (3) Power dynamics and decision-making processes significantly impact software testing outcomes, with project managers' decisions often dominating software testers' work. (4) Adhering to organizational processes and standards is essential for ensuring quality software delivery, preventing software testing from being bypassed or done hastily. (5) Legitimization of software testing practices is necessary for instilling social attachment and control among software testers, recognizing the importance of their role in delivering quality software. These findings highlight the significance of software testing in ensuring software quality and business continuity, emphasizing the need for effective communication, collaboration, and organizational processes to support software testers in their critical role.

KEYWORDS: Structuration Theory, Software testing, Software development, Software implementation, Software Development Life Cycle (SDLC), Information Systems

1. Introduction

In recent years, software testing has gained prominence in the software development industry [1]. Organisations of all sizes rely on software to deliver services and enhance productivity [2]. SDLC is a widely used methodology that outlines the stages of software development, from initiation to implementation [3]. Within SDLC, software testing is a critical phase that ensures software reliability and adds value to organisations [4]. A significant portion of software development budgets is allocated to testing [2]. Software testing encompasses various technical and non-technical sections, including specification, design, implementation, maintenance, and management issues [2]. Testing verifies that software meets organizational objectives and identifies errors or failures [5].

However, challenges like time constraints and regression testing hinder effective software testing [6]. This qualitative study employs structuration theory to analyze data and identify factors affecting software testing in organisations. The dynamics between social structures and human agency play an important role in shaping the adoption, use, and impact of information systems within organisations [7] [8]. Structuration Theory (ST), developed by Anthony Giddens, offers a valuable lens for examining these dynamics [7]. Despite its potential, ST has been underutilized in IS research [9], [10]. This study aims to address this gap by applying ST to explore the interplay between social structures and

human agency in the context of IS implementation [11]. Specifically, this research seeks to understand how social structures influence human agency and vice versa [12], and how this interplay affects IS outcomes [13]. By examining these dynamics, this study contributes to a deeper understanding of the complex factors that shape IS success and failure [14]. The paper is organized into five sections: literature review, research methodology, data analysis, findings, and conclusion.

2. Literature Review

This section covers existing literature in the following key areas of the study: (i) software development, (ii) Software testing, (iii) Software implementation, and (iv) Structuration theory.

2.1. The role of Software Testing

Software testing plays an essential role in ensuring the quality, consistency, and security of software products [4]. As software becomes increasingly pervasive in society, the significance of software testing cannot be overstated [15]. According to [16], software testing is a critical process that detects defects and ensures software meets user requirements. Effective software testing strategies enable organisations to identify and mitigate potential risks, reducing the likelihood of software failures and minimising their impact [17].

Any product that is created must be tested before it can be released to the general public for use or consumption. Same applies to any software that is developed by organisations to carry out their day-to-day duties. Software testing is the approach that guarantees that quality products are distributed to consumers, which in turn uplifts customer satisfaction and trust. The aim of software testing is to identify defects and issues in the software development process so that they can be fixed prior to its release. According to [18], it is vital to test software as it helps to verify its quality and reliability particularly in modern software development processes, where very sophisticated software is continuously released faster and quicker.

Even though software testing is important, its activities are usually ignored even by big organisations when executing significant software projects as they are often regarded unlikeable, time wasting as well as tedious when compared to more innovative and fulfilling activities such as software design or coding [19]. In [18], the authors defines software testing as "the process of evaluating software to ensure that it meets its originally specified requirements and revealing faults and defects that may affect the code". It verifies that the software meets the functional, performance, design as well as the implementation requirements identified in the functional requirement specification.

The primary intent of software testing is to guarantee that software functions as expected, meets user requirements, and is reliable, maintainable, and secure [20]. Software testing involves various activities, including test planning, test case development, test execution, and test reporting [21]. These activities guarantees that software is thoroughly vetted and meet the required standards before deployment [22].

Moreover, software testing is the fundamental component of the software development lifecycle, complementing activities such as system analysis, design, coding, and implementation [23]. Integrating software testing into the software development process, organisations can identify and report defects early, minimizing the overall cost and time needed for software development [24].

In conclusion, software testing is the important part of software development, guaranteeing that software meets the user requirements, are reliable, maintainable, and secure. By adopting effective software testing strategies, organisations can mitigate potential risks, reduce software failures, and release quality software products that meet the growing needs of society [25]. Lately, organisations, are focusing on software quality, and they are identifying broad requirements, such as more software functions, quicker response speed, as well as reliable and safe operation [26]. Software testing can be conducted in two main forms, either manually or automated. According to [27], manual testing can be time consuming, resource intensive and make testers not to discover some defects hence there are automation tools in place to enable both automation and performance engineers to record and rerun the test cases which could be tested manually by a software tester. Automated testing reduces the cost and time for testing software, it increases testing coverage by executing more test cases faster and eliminates human errors (humans gets tired when doing repetitive tasks and make errors) increased software reliability, user satisfaction and reduces the amount manual work that needs to be conducted by software testers. According to [27], tools for automating software testing enable software testers to consistently perform testing in less time and can frequently reuse them to retest the software. Automated testing decreases the volume of manual work, increases high coverage by executing additional test cases and reducing human

errors remarkably when humans are tired after several repeats [28]. In [29] further alluded that software testing is not performed only to detect defects but to assist software developers to notice the mistakes they made, provide tips on how to resolve those mistakes and also to ensure that the software performs as specified in the requirement specification.

2.2. System Development Life Cycle

Software development is a systematic process used to create software products that meet specific requirements and enable agents in a social system to achieve particular goals [25]. Organisations adopt software development strategies to manage software activities effectively and ensure alignment with business objectives [30]. According to [24] software development strategy refers to the approach organisations employ to develop the software. Software development encompasses various activities, including system analysis, design, coding, and testing [31]. Organisations consider the development of software as crucial for achieving business objectives and goals [25]. In [16], the author emphasizes the vital role of software in social systems, highlighting the need for extensive research, to understand, enhance, and support in software development.

2.3. Software Testing

Software testing is a crucial method that ensures software functions as expected, without defects or issues [4]. According to [16], software testing evaluates software quality and identifies areas for improvement. The primary goal of testing software is to discover defects and guarantee it meets user requirements [17]. However, it is essential to take into cognizance both functional and non-functional requirements during testing. Failure to do so may negative impact the quality of software [21]. For instance, the Gauteng online registration system failed to handle user load, despite functional testing [15]. Software testing tools enable testers to conduct both functional and non-functional testing, including performance testing, which determines software behavior under various conditions [20]. Automation tools enhance testing efficiency, reliability, and repeatability, reducing human error [20]. The ultimate goal of software testing is to deliver high-quality software, ensuring business confidence in the tested product [25]. Following testing, software implementation ensues, aiming to deploy the software for use.

The importance of software testing in the software development process cannot be overstated [4]. Software testing is crucial for ensuring the delivery of high-quality software products that meet user requirements and are reliable [17]. According to [16], software testing plays a vital role in identifying and fixing defects, errors, and bugs in the software, thereby reducing the likelihood of software failures and minimizing their impact.

Moreover, software testing helps save time and money by detecting defects early in the development process [24]. This is supported by [24], who argues that testing is an essential component of the software development lifecycle, complementing activities such as system analysis, design, coding, and implementation. Furthermore, software testing improves user experience by ensuring that the software meets user requirements [25]. It also enhances security by identifying security vulnerabilities and ensuring that the software is secure and protected against threats [15].

There are two main common types of software testing namely, black box and white box. The main of intent of black box testing is to test the behaviour of the software whereas white box testing focuses on testing the internal operation of the software. Black box testing also referred to as functional testing is a process whereby the software is tested without the knowledge of the internal workings of the software [32]. It is a method that enables the test engineer to design the test cases based on the information from the specification and does not allow the test engineer access to source code of the software [33] With black box, the test engineer is not required to have programming knowledge.

On the other hand, white box testing also referred to glass box testing/structural testing is a method that enables the test engineer/tester to design the test cases based on the information derived from source code [34]. The tester is required to have programming background with this type of testing as they are granted access to the source code. Grey box testing is a third method whereby the tester has limited knowledge about the internal workings of the software and has the knowledge of fundamental aspects of the software [35].

In addition, software testing supports continuous improvement by providing feedback for refinement and enhancement of the software [20]. This is critical for building trust with customers, stakeholders, and users, enhancing the organization's reputation [22]. In conclusion, software testing is essential for delivering high-quality software products that meet user requirements, are reliable, and provide a positive user experience [21]. By prioritizing software testing,

organisations can reduce risks, save time and money, and build trust with their customers.

2.4. Software Implementation

All software follows a particular lifecycle prior to its completion, from development to deployment, irrespective of the methodology employed (agile or traditional methodology), depending on the requirement [3]. Software implementation refers to the process of making software available for operation [1]. The term implementation is used interchangeably, but in the context of this study, it means making software operational. In the SDLC, implementation refers to applying system requirements, or actual coding [33]. Others describe it as constructing or building software [21], or making it available for use after development [7]. Software implementation occurs after quality assurance accompanied by various tests, including user acceptance testing [22]. Quality assurance is a challenging factor during implementation [1]. Top management approval is also crucial for successful implementation, as they must approve software before it is implemented or changed [7].

Software implementation poses several challenges that can hinder its success. One of the primary challenges is ensuring quality assurance, as inadequate testing can lead to software failures and errors [1]. Additionally, resistance to change from end-users can also pose a significant challenge, as they may be reluctant to adopt new software and processes [7]. Furthermore, the implementation of software needs substantial resources, including time, money, and personnel, which can be a challenge for organisations with limited budgets [33]. Moreover, integrating new software with existing systems and infrastructure can also be a complex challenge [23]. In [33], finally, top management approval and support are crucial for successful implementation, and lack of commitment from leadership can lead to implementation failure [22].

In conclusion, software testing and implementation are critical components of the SDLC. Effective software testing ensures that software meets user requirements, is reliable, and provides a positive user experience. However, software implementation poses several challenges, including quality assurance, resistance to change, resource constraints, integration with existing systems, and top management approval. To overcome these challenges, organisations must prioritize software testing and implementation, adopt effective software development methodologies, and ensure stakeholder commitment. By doing so, organisations can deliver high-quality software products that meet user needs and drive business success.

2.5. Structuration Theory

The selection of an appropriate theory to underpin a study is critically important because it assists in determining the outcomes of the study [11]. Structuration theory (ST) was developed by [8], it is a sociology theory but has also gained popularity in the information system (IS) field where it has been borrowed to analyse data [36]. The theory takes a stance that social action cannot be explained in detail through structure or agency alone, but it appreciates the actors operating within the context of rules shaped by social structures but act in a biddable manner that these structures reinforced. In [37], the authors defines ST "as the reproduction of social structures through human actions. Social structures and human actions are viewed as two aspects of the same whole, instead of seeing human actions happening outside of the constraints of social structure". According to [38], ST puts an emphasis on agency and structure, their duality within a social system which implies that the agent/agency entails technical such as technology and non-technical such as human entities. The structure is the rules and resource in structuration. In [39], the authors further states that ST focuses on how events and social systems are produced and reproduced over a period of time and space. In [40], the researchers alluded that human agency as well as social structure cannot be treated as separate ideas but are two ways of regarding social action and is termed as duality of structure. Structure is the recurrent patterned arrangements which influence or limit the choices and opportunities available. Whilst agency is the capacity of individuals to act independently and to make their own free choices [36]. ST is divided into structure, modality and interaction whereby the modality provides interaction between structure and interaction [37].

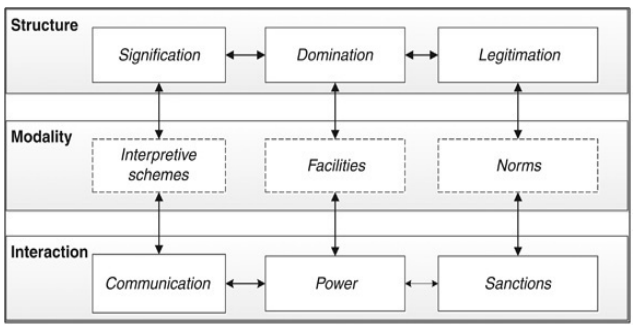

Figure 1: Dimensions of the duality of structure [26]

Both the social structures as well as the human actions are treated as two aspects of the same whole within the duality of structure [8]. In [40] and [41], the

authors emphasizes Giddens' claim that social phenomena emantes from both social structure and agency, not just one or the other. Figure 1 illustrates the dimensions of the duality of structure [8]. In IS research, various theories are used, including social theories [11]. IS researchers have borrowed b social theories, such as Actor-Network Theory (ANT), Activity Theory (AT), and Structuration Theory (ST) to underpin their studies [8].

This underscores the importance of software testing in ensuring that developed software meets requirements and is reliable, maintainable, and secure [4]. The SDLC comprises a sequence of connected methods from planning to system testing, ensures consistency and produces well-developed software [20]. As software becomes increasingly prevalent in society, the call for reliable, maintainable, and secure software requirements continues to grow [15].

3. Research Methodology

Qualitative approach was adopted for this study to gain an in-depth understanding of software testing in organisations from the participants' perspective [42]. Qualitative methods, as noted by [43], are subjective in nature, focusing on beliefs and encounters rather than statistical figures. A case study was used, which is usually employed in qualitative research [44]. This design allows for an in-depth examination of a phenomenon within its real-life setting [44]. The case study organization, Setlamo Technologies (a pseudonym), is a public sector organization operating in South Africa, with a dedicated software testing team. Semi-structured interviews were used to collect data, offering flexibility and allowing for clarifications during the interviews [45]. Therefore, this approach allows for in-depth examination without deviating from the research's core focus [46]. Fourteen participants were interviewed until saturation was reached [47].

The data was transcribed and analyzed using Structuration theory as a lens, focusing on the duality of structure [8]. The vertical approach was adopted, and an interpretivist approach was employed to analyze the findings [47]. The interpretive paradigm was used to subjectively interpret the findings [11].

This study adopted a qualitative case study approach, utilizing semi-structured interviews, to explore software testing practices in-depth [44]. This methodology is best suited for examining complex phenomena, such as software testing, in real-life settings [42]. Semi-structured interviews provide flexibility to explore topics in-depth, allowing participants to share experiences and opinions in their own words [45]. The case study method enables a comprehensive understanding of the organization, including social structures, human agency, and technology interactions [8]. This approach allows for a detailed examination of software testing practices, processes, and contextual factors, making it the most appropriate methodology for this study [48]. Moreover, identifying influential factors and potential adopters, as well as understanding their decision-making processes, is crucial for a thorough understanding of the subject [49].

4. Data Analysis

For the purpose of data analysis, participants and organisations were labeled. Fourteen employees from Setlamo Technologies participated in the research. The referencing standard is exemplified as ST01, 7:17-20, indicating organization ST, participant 01, page number 7, and line numbers 17 to 20.

i. Setlamo Technologies: Participants ST01 to ST14

At Setlamo Technologies, agents and structures were involved in software testing. These agents and structures were from the IT department. The agents comprised both technical and non-technical individuals. The structure entailed the rules and resources utilized in software testing. All participants shared similar interest for realizing organisational objectives through software testing and utilization.

4.1. Structuration Theory

According to [33], Structuration Theory (ST) encompasses agents (agency) and structures. Agents can be technical or non-technical, while structures consist of rules and resources involved in software testing activities within the organisation. The researchers identified agents followed by structures.

4.2. Agents (Agency)

Agency refers to an organisation comprising technical and non-technical agents, where some apply knowledge/conscious (human) and others lack knowledge capability [50]. Agency can also be associated to individuals or group of abilities within a particular environment. In the case of this study individuals are those who are involved in the SDLC such as software developers, software testers and others. They have particular skills to enable them to perform their duties. Therefore, these individuals can collaborate and apply their various skills to deliver a working solution within the organization (environment). An agent can be understood as anything with the potential to make a

difference in a social structure [51]. The research identified technical agents, including software testing tools such as Rational Quality Manager and Meter, as well as SAP, Oracle, and the environment comprising development, testing, and production. Additionally, technical agents included the Integrated Financial Management System (IFMS) and International Software Testing Qualifications Board (ISTQB) training.

Non-technical agents included software test analysts, project managers, business analysts, functional support personnel, software developers, and individuals involved in software development methodology, either Agile or traditional. Furthermore, non-technical agents comprised individuals responsible for software testing standards, implementation policies, and the change management committee. Documentation, including functional requirements, test cases, and test results, as well as the test lab, were also identified as non-technical agents.

4.3. Structure

The structure refers to the protocol followed to accomplish the tasks assigned to individuals within the team. Every individual has to take responsibility to delivering their tasks on scheduled time. Continuous reporting is necessary especial when an individual is struggling to perform what is assigned to them. These enables others to chip in and assist so that the project can be delivered on a promised time. Agents utilize structures to create and recreate social activities [34]. In the context of Structuration Theory, structure refers to rules and resources. Rules comprise regulations and policies guiding software testing activities, including the change management process and implementation policy. Resources encompass material and non-material objects used to carry out actions In this research, resources included software test analysts, project managers, business analysts, functional support personnel, software developers, software development methodology, software testing standards, implementation policies, change management committee, documentation (functional requirement, test cases, test results, test plan), and test lab [52].

4.4. Duality of Structure

4.4.1. Signification

Organisations develop or enhance existing software to render services, sell products, and conduct day-to-day activities. This software must be rigorously tested prior to implementation to eliminate defects. Failure to do so may result in losing existing clients and failing to attract potential clients. One participant emphasized that *"Testers uncover and eliminate things that we designers and analysts have overlooked when we were planning"* (ST_06, 17:647-648). Software testing ensures business continuity and quality, as highlighted by a functional support personnel: *"We want to deliver a quality and functional software that meets the user requirements"* (ST_11, 27:1041). Interpretive Scheme (Stock of Knowledge)

It was vital for other employees within the organization to understand the significance of software testing. However, employees from other departments perceived software testing as a waste of time and a delay in implementation. This lack of understanding developed a negative perception towards software testing. One participant noted that *"People who are not involved in software testing think that software testing is not important"* (ST_01, 06:215). Consequently, only the software testing team comprehended the value of software testing, as stated by a software tester: *"Maybe do some roadshow and explain to them what exactly testing entails in order for them to get a broader picture of what testing is"* (ST_03, 08:312-313).

Even a software developer expressed uncertainty about the role of software testing: *"I do have an idea but I don't really know what they do...let's say after unit testing I know that we hand over the software to them...they already have some test cases, test scripts or whatever...mostly they will be verifying characters and send the results to me"* (ST_13, 29:1142-1144). This lack of understanding created animosity between non-technical agents (project team members).

4.4.2. Communication

Effective communication was crucial among project team members to deliver quality software. Both functional and non-functional requirements were communicated through the functional requirements specifications. Verbal communication among team members and updates to the functional requirement specification ensured clarity. A software tester emphasized that *"Others felt that by verifying software we are judging that they did not do their work properly...especially when a tester logs a defect against the developer...they will argue with you and even fight with you verbally"* (ST_03, 08-09:316-319).

Software testers relied on the functional requirement specifications to verify the validity of the developed software. The developed software needed to correspond with the specified documents to deliver quality software. Therefore, within this organization, requirements were communicated through documentation, such as business

requirements, functional specifications, and technical requirement specifications: *"All the user requirements are documented, and the user requirement specification is signed off prior to development"* (ST_05, 12:472-473).

4.4.3. Power

Solid decisions needed to be made regarding software development, testing, and implementation. The change management committee held the power to decide whether software development was necessary. They relied on testing results from the software testing team to decide whether software was ready for implementation. A functional support personnel stated that *"We have a change management department where they actually decide whether the change is needed or not"* (ST_07, 18:674-675).

Other employees within the organization exercised their power to influence project prioritization. This power was influenced by the position (facility) individuals occupied within the organization, such as the CIO. When the CIO committed to a project, it dominated other projects and received high priority. A test analyst noted that *"The CIO committed to that project, and they drilled down to the testing team, and the project was really regarded as important, and it had to undergo testing"* (ST_08, 21:797-799).

4.4.4. Facility

The change management committee exercised power over the agency (software development team) based on the facility (authority) granted to them by the organization. They made decisions about what needed to be developed, tested, and implemented. Consequently, they relied on software testing results to decide whether software was implemented or not: *"We have a change management department where they actually decide whether the change is needed or not"* (ST_07, 18:674-675).

On the other hand, project managers had the tendency to decide on behalf of the software testing team. As a result, project managers dominated other teams, such as the software testing team, by imposing timelines without consulting the team. A software tester noted that *"Unrealistic schedules from the project team...because if the project is not scheduled properly, it puts testing under pressure...for example, testing could be allocated three months for conducting all the testing, which makes it difficult for the testing team to meet timelines"* (ST_08, 20:781-783).

4.4.5. Sanction

The organisation had processes in place that needed to be followed to implement software. However, instances occurred where these processes were bypassed by some employees, becoming a norm for many projects. Consequently, many software failures occurred in production. A participant stated that *"Processes are not followed at all, and management is doing nothing about the situation"* (ST_05, 14:545).

According to the organisation's regulations, software should be developed, tested, and then implemented. This regulation should be the norm for the agency (software development team). However, the change management committee needed testing results to decide whether to implement software.

4.4.6. Norm

Adhering to standards, procedures, and policies became a norm for employees within the organization. However, instances occurred where project managers prepared project schedules and made estimations without consulting the testing team. Also, project managers promised to deliver software to business within a particular duration that was not agreed upon with the software testing team. As a result, the software testing team was pressed to complete testing within a short period, leading to working overtime and even coming in on weekends to finish their work. Such working conditions negatively impacted software quality, as a tired software tester was more likely to make mistakes.

The relationship between software testers and software developers was strained due to logged defects. Software developers felt that software testers were not recognizing their hard work. Some software was implemented without being tested and was tested in production because scheduled timelines were not met. The test manager asserted that *"Quality will be impacted, and as a result, you are likely to produce production issues when you deploy any application or system that didn't follow a proper process"* (ST_14, 34:1317-1318).

As a result, it became a norm not to follow proper processes when testing software within the organization. Therefore, it was management's responsibility to enforce standards, procedures, and policies. The software testing team complained to their test manager, who alerted management about processes not being followed by other members of the software development team. However, nothing seemed to be changing. Thus, one software test analyst angrily stated that *"Processes are not followed at all, and management is doing nothing about the situation"* (ST_05, 14:545).

It was vital for employees to follow organizational processes to deliver quality software. Consequently,

Setlamo Technologies' clients would be satisfied with what was delivered to them. One participant stressed that *"It is important to make sure that processes are followed because, in that case, testing will not be bypassed"* (ST_04, 11:427-428).

4.4.7. Legitimation

Irrespective of either right or wrong, a norm is legitimized. Thus, it is important to do things properly within the organization. If nothing is done about it, then the organization would fail to achieve its goals, and the client would be unhappy with the quality of software delivered. Therefore, some agents in the organization legitimize software testing practices by accepting it as the norm. These agents understood that once software is developed, it needs to be tested to ensure it meets client requirements. A software tester noted that *"Testing is a crucial part of the software development life cycle...you cannot just develop and implement without testing"* (ST_02, 07:263-264). This legitimation of software testing practices is essential for the organization to deliver quality software to its clients. However, some agents within the organization did not legitimize software testing practices, leading to software testing being bypassed or not being done properly.

4.4.8. Domination

The power dynamics within the organization led to domination by some agents over others. Project managers dominated software testers by imposing timelines without consulting them. This domination led to software testing being done hastily, resulting in poor quality software being delivered to clients. A software tester stated that *"Project managers promise to deliver software to the business within a particular duration that was not agreed upon with the testing team"* (ST_08, 20:781-783).

4.4.9. Signification

Software testing signified quality software delivery to clients. It ensured that software met client requirements and was free from defects [28]. A software tester emphasized that *"Testing ensures that the software meets the requirements...it ensures that the software is functional and works as expected*" (ST_06, 17:647-648). However, some agents within the organization did not signify software testing, leading to poor quality software being delivered to clients.

4.5. Software Testing in the SDLC: A Structuration Theory Perspective

At Setlamo Technologies, software testing is a critical component of the Software Development Life Cycle (SDLC). Our research identified agents (technical and non-technical) and structures (rules and resources) involved in software testing [29]. Technical agents included testing tools and methodologies, while non-technical agents comprised project managers, business analysts, and software developers.

4.6. Current SDLC Market Trends

- Agile methodologies emphasize collaboration and communication among agents.
- DevOps practices integrate testing into the development process.
- Continuous Testing and Continuous Integration/Continuous Deployment (CI/CD) pipelines automate testing processes.
- Artificial Intelligence (AI) and Machine Learning (ML) enhance testing efficiency and effectiveness.

Agile methods vary in their ways, but they share a common aim which is to enable their teams swiftly respond to change [53]. In [54] stated that when modifications are expensive to adjust to later in the project, the capability to respond quicker to modification minimizes the project risks and their budgets [55]. In [56] alluded that while agile methods are efficient, huge, and complex software products often needs methodical discipline with the obligatory process to guarantee success. On the other hand, DevOps was introduced around 2007 and 2008 after software development communities realized the fatal dysfunction within the software development landscape. There was a disconnect between those who develop the software and those who implement and maintain the software. According to [62] often in the software deployment, employees who are involved in the development of software are not necessarily the ones who are involved in the implementation, hence the disconnect is encountered. DevOps approach assists in delivering value faster and uninterruptedly, minimizing challenges because of miscommunication between team members as well as fast-tracking problem resolution [57]. In [58], alluded that DevOps is an organisational shift which substitute distributed siloed groups executing tasks separately with cross-functional teams which work on continuous operational feature deliveries. In simple terms DevOps is a culture shift which provides collaboration amongst development, quality assurance and operations. In [59] the authors highlighted that while continuous integration (CI) combines work-in-progress numerous times a day, continuous deployment (CD) focuses on to possible release values to consumers faster and capably by employing automation as much as possible. Artificial

Intelligence (AI) and Machine Learning (ML) are not just a buzzword in the digital era but the best way of doing things. Gone are those times of doing things in a traditional manner. AI makes provision for countless improved results on a bigger scale and more complex neural networks, packed with many layers deep learning and much progress can be ascribed to bigger data sets and large-scale learning/training on graphic processing unit [60]. Computers are taught to emulate humans through performing complex tasks which used to be historically performed by humans such as reasoning, making decisions or solving problems. Whilst ML which is a subset of AI which is used to learn from large data sets. It enables computers to learn from data without being explicitly programmed [61]. DL on the other hand enables computers to learn complex concepts through creating them out of simpler ones [62]. It uses neural networks to process data like humans.

Our research highlights the importance of software testing in the SDLC, emphasizing the need for effective communication, collaboration, and process adherence. As the SDLC market continues to evolve, organisations must prioritize software testing to deliver high-quality products and services. By embracing current trends and best practices, organisations can optimize their software testing processes and stay competitive in the market.

5. Findings

Poor management and lack of process compliance were significant factors contributing to poor quality software at Setlamo Technologies. The organisation's failure to enforce processes, policies, standards, and procedures led to software testing being treated as an afterthought, resulting in poor quality software. This lack of emphasis on software testing also led to a culture of neglect, where software testing was seen as a mere formality rather than a critical aspect of software development.

The software testing team faced unrealistic timelines, and their concerns were ignored by management, leading to frustration and high turnover rates. This highlights the need for management to prioritize software testing and provide adequate resources and support to the testing team. Non-compliance to processes was a norm, with some employees bypassing the change management committee and implementing software without testing. This lack of compliance led to poor quality software, reputational damage, and loss of customers. It also suggests a lack of accountability and a culture of siloed work, where individuals prioritize their own goals over the organization's overall objectives.

Furthermore, a lack of software testing knowledge among employees contributed to the undervaluing of software testing, leading to frustration among software testers and a high turnover rate. This highlights the need for training and education programs to ensure that all employees understand the importance and benefits of software testing. Furthermore, a lack of software testing knowledge among employees contributed to the undervaluing of software testing, leading to frustration among software testers and a high turnover rate. This highlights the need for training and education programs to ensure that all employees understand the importance and benefits of software testing.

The disconnect between project stakeholders, including project managers, software developers, and testers, also hindered effective software development and testing. This suggests a need for improved communication, collaboration, and integration among stakeholders to ensure that software development and testing are aligned with organizational goals.

To address these issues, management must enforce processes, policies, standards, and procedures, and ensure that all employees understand the value of software testing. Additionally, stakeholders must work collaboratively to deliver good quality software, and management must address non-compliance and knowledge gaps to prevent poor quality software. This may involve implementing quality control measures, providing training and education programs, and fostering a culture of collaboration and accountability.

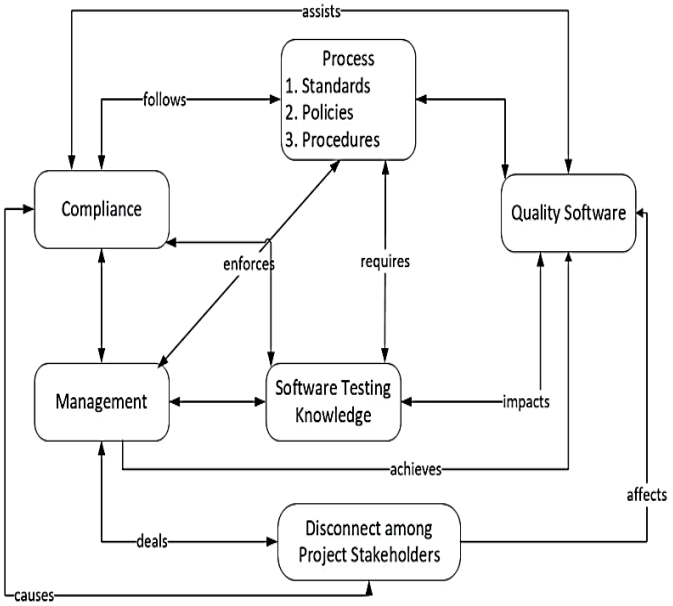

Figure 2: Factors affecting the quality of software at Setlamo Technologies

6. Summary

In summary, the study reveals significant challenges in software testing practices at Setlamo Technologies, including:

6.1. Poor Management and Lack of Adherence to Processes, Policies, and Standards

Poor management and lack of adherence to established processes, policies, and standards were significant contributors to the software quality issues at Mmuso Technologies. The absence of effective leadership and clear goals led to a lack of direction and focus among team members. Furthermore, the failure to enforce processes, policies, and standards resulted in a culture of non-compliance, where employees bypassed established procedures, leading to inconsistent and poor-quality software development.

6.2. Non-Compliance to Processes, Leading to Untested Software Implementation

Non-compliance to processes led to untested software implementation, which significantly impacted software quality. The lack of adherence to established processes resulted in software being implemented without proper testing, leading to errors, bugs, and defects. This not only affected the software's performance but also compromised its reliability and security. The failure to follow established testing processes led to a lack of confidence in the software's quality, ultimately affecting customer satisfaction.

6.3. Poor Quality Software, Resulting from Inadequate Testing and Lack of Cooperation among Stakeholders

Poor quality software was a direct result of inadequate testing and lack of cooperation among stakeholders. Insufficient testing led to undetected errors, bugs, and defects, while the lack of cooperation among stakeholders hindered effective communication, collaboration, and coordination. This resulted in software that failed to meet customer requirements, was unreliable, and lacked security. The absence of a collaborative environment led to a lack of accountability, further exacerbating software quality issues.

6.4. Lack of Software Testing Knowledge among Employees, Leading to Frustration and Undervaluation of Testing

The lack of software testing knowledge among employees led to frustration and undervaluation of testing. Employees without proper training and understanding of testing principles and methodologies struggled to effectively test software, leading to inadequate testing and poor software quality. The undervaluation of testing resulted in a lack of resources, support, and recognition for testing efforts, further demotivating employees and perpetuating software quality issues.

6.5. Disconnect Between Project Stakeholders, Causing Process Non-Compliance and Poor Software Quality

The disconnect between project stakeholders led to process non-compliance and poor software quality. Poor communication, collaboration, and coordination among stakeholders resulted in a lack of understanding of project requirements, leading to non-compliance with established processes. This, in turn, led to poor software quality, as stakeholders worked in silos, prioritizing individual goals over project objectives. The absence of a unified approach led to a lack of accountability, further exacerbating software quality issues. These challenges lead to reputational damage, customer loss, and decreased trust in the organization. To address these issues, the organization must:

1. Enforce processes, policies, and standards.
2. Educate employees on software testing's value.
3. Foster cooperation and communication among stakeholders.
4. Address knowledge gaps and provide training.
5. Encourage a culture of quality and testing.

By addressing these challenges, Setlamo Technologies can improve software quality, increase customer satisfaction, and maintain a competitive edge in the industry.

6.6. Key Findings:

Software testing is crucial for delivering quality products and services, ensuring that software meets client requirements and is free from defects.

1. Effective communication and collaboration among agents, including software testers, developers, and project managers, are vital for successful software testing outcomes.
2. Power dynamics and decision-making processes significantly impact software testing outcomes, with project managers' decisions often dominating software testers' work.
3. Adhering to organizational processes and standards is essential for ensuring quality software delivery, preventing software testing from being bypassed or done hastily.
4. Legitimization of software testing practices is necessary for instilling a sense of belonging and

control among software testers, recognizing the importance of their role in delivering quality software.

7. Recommendations

1. Establish clear communication channels and collaboration frameworks to facilitate effective interaction among agents involved in software testing.
2. Empower software testers by involving them in decision-making processes and providing autonomy in their work to ensure quality software delivery.
3. Develop and enforce organizational processes and standards that prioritize software testing, preventing domination by project managers' decisions.
4. Provide training and resources to software testers to enhance their skills and knowledge, legitimizing their role in delivering quality software.
5. Conduct regular assessments and evaluations to identify areas for improvement in software testing practices, ensuring continuous quality improvement.

8. Conclusion

In conclusion, this study highlights the critical role of software testing in delivering quality products and services, emphasizing the need for effective communication, collaboration, and empowerment of software testers. The findings underscore the impact of power dynamics and decision-making processes on software testing outcomes, stressing the importance of adhering to organizational processes and standards. By implementing the recommended measures, organisations can legitimize software testing practices, foster a sense of belonging and control among software testers, and ultimately ensure the delivery of high-quality software that meets client requirements. Moreover, this study demonstrates that addressing the challenges in software testing practices can have far-reaching benefits, including:

- Enhanced software development life cycle
- Improved customer satisfaction
- Increased trust and reputation
- Better decision-making processes
- Empowered software testers
- Competitive edge in the industry

By prioritizing software testing and addressing the identified challenges, organisations can unlock these benefits and deliver high-quality software products and services that meet the evolving needs of their clients. Ultimately, this study contributes to the growing body of knowledge on software testing practices, emphasizing the need for a collaborative, empowered, and process-driven approach to software testing.

Acknowledgement

We extend our heartfelt appreciation to ESKOM South Africa, Department of Research, for their generous sponsorship, which has enabled us to publish this paper. We are deeply grateful for their recognition of the significance of research and its contribution to the existing body of knowledge. Their support and understanding are truly valued, and we express our sincere thanks.

References

[1] T. Bryant, Software Development: A Practitioner's Approach, Routledge, 2017.

[2] R. Tuteja and S. K. Dubey, Software Testing: Concepts and Operations, PHI Learning, 2012.

[3] J. P. Kotter, "Leading Change," *Harvard Business Review Press,* 2012.

[4] G. J. Myers, "The Art of Software Testing," *John Wiley & Sons,* 2011.

[5] M. Oluigbo, L. Erasmus, and R. Snyman, "An Exploratory Study of Software Testing Practices in South Africa," *South African Computer Journal,*, vol. 29, no. 1, 1-15, 2017.

[6] J. Cameron and P. Green, Software Testing: A Guide to the TMap Approach, Pearson Education,, 2015.

[7] A. Giddens, Central Problems in Social Theory: Action, Structure, and Contradiction in Social Analysis, University of California Press, 1979.

[8] W. J. Orlikowski, "The Duality of Technology: Rethinking the Concept of Technology in Organizations", Organization Science," vol. 3, 398-427, 1992.

[9] C. Jones, "Software Project Management Practices: Failure to Apply Project Management Principles," 2011.

[10] M. Pozzebon, "The Influence of a Quality Management System on the Software Development Process," *Journal of Systems and Software,* 2004.

[11] H. K. Klein, M. D. Myers, "A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems," *MIS Quarterly,* 2011.

[12] W. H. Sewell, "A Theory of Structure: Duality, Agency, and Transformation," *American Journal of Sociology,* vol. 98, no. 1, pp. 1-29 , 1992.

[13] W. J. Orlikowski, "Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations. Organization Science"," *Organization Science,* vol. 11, 404-428, 2000.

[14] G. Walsham, "Interpreting Information Systems in Organizations," *John Wiley & Sons,* 1993.

[15] I. I. IEEE29119-1:2018, "Software and Systems Engineering — Software Testing — Part 1: Concepts and Definitions," *International Organization for Standardization,* 2018.

[16] E. Dustin., "Automated Software Testing: A Guide for Software Project Managers," *Charles River Media,* 2017.

[17] C. Kaner, Lessons Learned in Software Testing: A Context-Driven Approach, John Wiley & Sons, 2013.

[18] T. Fulcini et al., "A review on tools, mechanics, benefits, and challenges of gamified software testing," *ACM Computing Surveys,* vol. 55, no. 14s, 1-37, 2023.

[19] D. Deak et al., "The Impact of Agile Methods on Software Project Management," *International Journal of Information Technology Project Management,* 2016.

[20] IEEE, "IEEE Standard for Software and System Test Documentation (IEEE Std 829-2019)," *IEEE Computer Society,* 2019.

[21] P. E. Black, "Managing Software Projects. In Encyclopedia of Software Engineering," *CRC Press* , 359-373, 2008.

[22] ITIL, ITIL Foundation: ITIL 4 Edition, AXELOS, 2019.

[23] I. Sommerville, Software Engineering, Pearson, 2016.

[24] R. S. Pressman, "Software Engineering: A Practitioner's Approach," *McGraw-Hill,* vol. 2, 41-42., 2010.

[25] K. Laudon, J. P. Laudon, Management Information Systems: Managing the Digital Firm, Pearson, 2015.

[26] Y. Zhao et al., "Software Quality Requirements in the Context of Digital Transformation," *International Journal of Software Engineering and Knowledge Engineering,* 2021.

[27] T. Sekgweleo, T. Iyamu, "Software testing: some influencing factors in a South African organisation," *Journal of Contemporary Management,* vol. 17, no. 1, 86-107, 2020.

[28] O. Ibitomi et al., "Automation of Software Testing: A Systematic Review," *Journal of Software Engineering and Applications,* vol. 17, no. 1, 1-22, 2021.

[29] T. Sekgweleo, "Disjoint between development and deployment of software," (Masters dissertation, Tshwane University of Technology, 2011).

[30] G. Bansal, "Software Development Strategy, In Encyclopedia of Software Engineering," *Taylor & Francis,* 1-10, 2008.

[31] G. Ghosh, "Software Development: Principles, Methodologies, Tools, and Techniques," *CRC Press,* 2017.

[32] T. G. Sekgweleo, "A decision support system framework for testing and evaluating software in organisations," (Doctoral dissertation, Cape Peninsula University of Technology, 2018).

[33] K. Avison, G. Fitzgerald, Information Systems Development: Methodologies, Techniques and Tools, Pearson, 2015.

[34] S. Nidhra, J. Dondeti, P. Katikar and S. Tekkali, "Implementing the concept of refactoring in software development," *In 2012 CSI Sixth International Conference on Software Engineering (CONSEG),* 1-8, 2012.

[35] M.E. Khan, F. Khan, "A comparative study of white box, black box and grey box testing techniques," *International Journal of Advanced Computer Science and Applications,* vol. 3, no. 6, 1-141, 2012.

[36] T. Sekgweleo et al., "Structuration Theory: A Review of the Literature," *Journal of Sociology and Social Anthropology,* vol. 8, no. 2, 147-164, 2017.

[37] T.G. Sekgweleo, M. Makovhololo, "Structuration Theory: A Framework for Understanding Software Testing," *Journal of Software Engineering and Applications,*, vol. 16, no. 1, 1-15, 2023.

[38] T. Iyamu, D. Roode, "The use of structuration theory and actor network theory for analysis: Case study of a financial institution in South Africa," *Social influences on information and communication technology innovations, IGI Global,* 1-19, 2012.

[39] L. Ma, Knowing and teaching elementary mathematics: Teachers' understanding of fundamental mathematics in China and the United States, Routledge, 2010.

[40] B. P. Lamsal, "Production, health aspects and potential food uses of dairy prebiotic galactooligosaccharides," *Journal of the Science of Food and Agriculture* , vol. 9, no. 10, 2020-2028, 2012.

[41] W. H. Sewell. Jr, "A theory of structure: Duality, agency, and transformation," *American journal of sociology,* vol. 98, no. 1, 1-29, 1992.

[42] J. W. Creswell, "Research Design: Qualitative, Quantitative, and Mixed Methods Approaches," *Sage Publications,* 2014.

[43] M. Q. Patton, "Qualitative Research and Evaluation Methods," *Sage Publications,* 2002.

[44] L. Rademaker, "Qualitative Research from Start to Finish: A Book Review," *Qualitative Research,* vol. 16, no. 5, 1425-1428, 2011.

[45] S. Kvale, "Interviews: Learning the craft of qualitative research interviewing," *Sage,* 2009.

[46] P. Nemutanzhela, T. Iyamu, "A framework for enhancing the information systems innovation: using competitive intelligence," *Electronic Journal of Information Systems Evaluation,* vol. 14, no. 2, 242-253, 2011.

[47] J. Low, "A pragmatic definition of the concept of theoretical saturation," *Sociological focus,* vol. 52, no. 2, pp. 131-139, 2019.

[48] G. Walsham, "Decentralization of IS in developing countries: power to the people?," *Journal of Information Technology,* vol. 8, no. 2, 74-81, 1993.

[49] P. Makovholo et al., "Diffusion of innovation theory for information technology decision making in organisational strategy," *Journal of Contemporary Management,* vol. 14, no. 1, 461-481, 2017.

[50] Y. Sarason et al.,, "Entrepreneurship as the nexus of individual and opportunity: A structuration view," *Journal of business venturing,* vol. 21, no. 3, 286-305., 2006.

[51] M. Peillon, "The Constitution of Society, Outline of the Theory of Structuration," *Oxford University Press,* vol. 1, no. 3, 261-263, 1985.

[52] N. Barqawi, "Software service innovation: an action research into release cycle management," 2014.

[53] M. Coram, S. Bohner, "The impact of agile methods on software project management," *In 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05),* 363-370, 2005.

[54] F. Paetsch et al., "Requirements engineering and agile software development," 2003.

[55] K. Beck, eXtreme Programming Explained, Addison-Wesley, 2000.

[56] T. Sekgweleo, T. Iyamu, "Empirically Examined the Disjoint in Software Deployment: A Case of Telecommunication," *International Journal of Actor-Network Theory and Technological Innovation,* vol. 4, no. 3, 36-50, 2012.

[57] M. Virmani, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery," 2015.

[58] M. Standar, "Continuous architecture in a large distributed agile organization: A case study at Ericsson," *IEEE Explore,* vol. 33, no. 3, 1-104, 2017.

[59] R. T. Yarlagadda, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery," *International Journal of Emerging Technologies and Innovative Research,* 2349-5162, 2018.

[60] R. Feldt et al., "Ways of applying artificial intelligence in software engineering," 2018.

[61] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR,* vol. 9, no. 1, 381-386, 2020.

[62] M. R. Minar, J. Naher, "Recent advances in deep learning: An overview," *arXiv preprint arXiv:1807.08169,* 1-31, 2018.

Tefo Gordon Sekgweleo has done his master's degree from Tshwane University of Technology in 2012. He completed his PhD degree from Cape Peninsula University of Technology in 2018. He started his career as a software developer, he moved to software testing as a a software automation engineer. He became a software testing manager, I have 36 publications, and currently working as a research manager for digitalization.

Phathutshedzo Makovhololo is a distinguished IT professional and scholar with 18 years of experience in leadership and senior management roles. Holding a PhD in Informatics, she possesses expertise in IT governance, policy management, business analysis, and project management. With a strong ability to bridge the gap between technology and business strategy, Dr. Makovhololo has a proven track record of effective leadership, lecturing, and research. Her notable strengths include visionary leadership, excellent communication, and strategic thinking, complemented by a strong research and analytical skillset.