Review Article

Insights in Science and Technology

Application of Actor Network Theory and Diffusion of Innovation in Evaluating and Testing Software: Case of Public Company

Dr. Tefo Gordon Sekgweleo*

Eskom Research, Testing, Development, South Africa.
ORCID: 0000-0002-5216-6892

***Correspondence:**
Dr. Tefo Gordon Sekgweleo, Eskom Research, Testing, Development, South Africa.

Received: 02 May 2026; **Accepted:** 03 Jun 2026; **Published:** 14 Jun 2026

Citation: Tefo Gordon Sekgweleo. Application of Actor Network Theory and Diffusion of Innovation in Evaluating and Testing Software: Case of Public Company. Insights Sci Technol. 2026; 1(1): 1-11.

ABSTRACT

The aim of software testing is to evaluate and verify that software functions correctly, meets specific requirements and is free of defects. It is a critical phase of the Software Development Life Cycle (SDLC) which assists in improving quality, reliability as well as security while reducing the long-term cost of fixing bugs. It promotes collaboration with various stakeholders involved in the systems development life cycle. This study focused on how software is evaluated and tested in a public company. Exploratory research approach was adopted, and data was collected through semi-structured interviews. Four moments of translation of ANT and communication channels of DoI was used as a lens to analyse the empirical data that was collected. Findings were explained accordingly to how software is evaluated and tested within the public company.

Keywords

Actor Network Theory (ANT), Black Box Testing, Diffusion of Innovation (DoI), Grey Box Testing, White Box Testing.

Introduction

There are various approaches when evaluating and testing software within the software development methodologies. They include black box, white box as well as grey box testing. Black box testing approach is the tester evaluates the functionality of the software without knowing its internal code structure. Testers focus exclusively on inputs and outputs, verifying that the software functions exactly as expected from an end-user perspective. In this approach, Testers provide inputs and verify outputs against expected results, without considering internal logic. Test cases are created based on requirements, user stories as well as the use cases to ensure full coverage and traceability. White box testing focuses on evaluating and testing the internal structure, design and code of the software. It is known as clear, glass, or structural testing as it provides testers with full visibility into the source code to verify that internal operations and logic work correctly. It ensures that the code logic, flow as well as the implementation functions correctly as per requirements. Grey/Gray box testing combines the strengths of both black box and white box testing. It leverages partial knowledge of the internal structure while focusing on the system's functionality and behaviour from an external perspective.

Literature Review

Black Box Testing

Black box method is concerned with examining the functionality of the software without looking into its internal workings of the software. Black box testing is performed to compare the actual functionality of the software with the intended functionality described in the software specification document [1]. Mainly conducted to test the behaviour of the software, this method is divided into various techniques which includes equivalence partitioning, boundary value analysis testing and decision table testing [2]. Other black box testing techniques are the cause-effect graphing techniques, comparison testing, fuzz testing and model-based testing [3]. Table 1 describes the various black box techniques:

During black box testing, some parts of the software, especially the back end, are not tested at all [5], allowing the testing team to actually perform tests on only a selected number of test scenarios which leads to limited coverage [6] as a result of the lack of

Table 1: Black box techniques [4].

Equivalence Partitioning	This technique divides the input domain of a program into equivalence classes, a set of valid or invalid states for input conditions.
Boundary Value Analysis Testing	It is a positive or negative test focused on boundary or limit conditions of the software being tested.
Decision Table Testing	It is a test designed to execute the combinations of inputs based on conditions shown in the decision table.
Comparison Testing	It is a technique whereby the software engineering teams produce independent versions of the system and each version is tested with the same test data, so the same output can be confirmed.
Fuzz Testing	It is a degree to which a software can function correctly in the presence of invalid inputs.
Model-Based Testing	It is an automatic generation of efficient test procedures using models of system requirements and specified functionality.

Table 2: White box techniques [6].

Control Flow/Coverage Testing	It uses the flow of the program as a model to control flow and favours simpler paths over less but complicated paths.
Basic Path Testing	It ensures that each possible outcome from the condition is tested at least once.
Loop Testing	It exclusively focuses on the validity of loop construct.
Data Flow Testing	It ensures that the control flow graph has the information about how the program variables are defined and used.
Branch Testing	It ensures that every option (true or false) is tested on every control statement, including compound decisions.

knowledge of internal workings of the software. Aichernig [7] contends that the black box approach does not consider how the test-object is implemented but considers what its requirements are. Some important parts of the tested software may be easily overlooked. The alternative to the black box testing is an approach known as the white box testing method.

White Box Testing

The white box method is concerned with testing software by examining the internal workings of the software. It requires the tester to possess the internal knowledge of the software as well as the programming skills. It is typically effective in validating design, decision, assumptions and finding programming errors and implementation errors in the software [8]. The testing is based on the code coverage, paths, branches and conditions. The intention is not to find every software defect that exists but to expose situations that could negatively impact the customer. This method is divided into various techniques: control flow/coverage testing, basic path testing, loop testing and data flow testing [9]. Table 2 below describes the above-mentioned white box techniques.

It is wise to begin testing early in the system development life cycle. Steegmans et al., [10] state that the biggest limitation of white box testing is that test suits can only be developed late in the life cycle of a software component. And beginning such testing late in the development life cycle affects testing timelines negatively as it leads to lapsing deadlines. As a result, even more time is required, increases the testing costs. There are, however, software testing tools in place that save time and cost. White box requires intimate knowledge of a target system, testing tools and coding languages and modelling [11]. These testing methods can be used in conjunction with various software testing tools. Another testing method that can be used is the grey box method.

Grey Box Testing

The grey box method, a combination of black box and white box testing, is a method used to test the software with limited knowledge of the internal workings of the system [12]. Moreover, it is the testing approach used when some knowledge of internal structure is known, but not in detail. Saxena and Singh [13] argue that the purpose of grey box testing is to examine if there is any defect due to improper structure or usage of the software. According to Bhasin and Kumar [14], this method is well suited for web applications, web services, functional or business domain testing, security assessment, GUI and distributed environments. Below, Table 3 describes the above-mentioned grey box techniques:

Some parts of the software may be missed due to limited access to internal workings of the software, resulting in partial code coverage [13]. This, however, defeats the purpose of end-to-end testing. The aim of software testing is to cover most parts of the software as much as possible. Due to restricted knowledge of the tester, it is not feasible to cover every part of the software with the situation that many program paths go untested. According to Archarya and Pandya, grey box testing continues to rely on how well the software throws exceptions and how well these exceptions are spread within a distributed web service environment. The tester relies on how the software reacts. These testing methods can be used in conjunction with various testing tools.

Research Methodology

Qualitative and exploratory research approach was adopted for this study. The intention of exploratory research is to formulate a problem to precisely investigate or develop working hypotheses from an operational point of view [15]. It allows people to think, to use their imagination, experience, insight and skill to propose innovative ways for understanding and interpreting reality [16]. A public company was used as a case study and data was collected through semi-structured interviews. Data was analysed using the moments of translation of ANT and the communication channels of DoI.

Analysis

Black box methodology, one of the tenants of the actor network theory, is described by Lihosit [17] as something that is not easy to understand or explain. For instance, while not much is known about new technologies, people still eagerly adopt these technologies, assuming they can do what they want. Therefore, this new technology could be regarded as a black box. Both ANT and software testing consists of black box testing. In ANT, black box is something that is not easy to understand or explain. In software testing, black box is when the software under test is tested without regards for the internal code structure, implementation details and knowledge of internal paths of the software. The similarity between the black box within ANT and software testing is that the internal workings of the software or technology are unknown. However, in ANT, this black box needs to be open to gain understanding,

Table 3: Grey box techniques.

Matrix Testing	The software developer begins by defining all the variables that exist in their programs and each variable has an inherent technical risk.
Regression Testing	This testing is performed after making a functional improvement or repair to the program to ensure that what was fixed have not affected other aspects of the program.
Pattern Testing	It helps to dig within the code and determines why the failure has happened.
Orthogonal Array Testing	It is a statistical testing technique that is extremely valuable for testing complex applications.

whereas in software testing there is no need to open the black box because the purpose is to test the functionality of the software, so it is not important to know the internal workings of the software but rather to ensure that the software behaves as expected.

Innovation is the new idea that is developed to be adopted by the social system. The rate of adoption is measured according to how it is accepted within the social system. According to Alqahtani and Wamba [18], DOI theory investigates how, why and at what rate, new ideas or technologies spread through cultures. Culture is the way in which people live, behave and do things within a particular environment. As a result, the key elements in diffusion of innovation come into play. Montfort et al., [19] describe diffusion as the procedure by which innovation is transferred through certain channels over time among the social system. Below is a diagram portraying various tenets of DOI. ANT was first applied in the analysis of data collected from the public company, followed by DOI.

Moments of translation: problemitization

It is the responsibility of the public company is to develop software for other government departments. Therefore, when any government department needed the software, they logged a request with public company. This software was developed to enable the government departments to function efficiently and to render services to the citizens of South Africa. To accomplish this, software had to be rigorously tested and evaluated to ensure that it functioned as expected. There were instances whereby the public company collaborated with external organisations and vendors to develop and configure other software, primarily due to lack of skills within the organisation. One participant alluded that: "We wanted to have a total independent software testing team which will verify the work of the vendors".

Due to the absence of independent software testing, the testing that was conducted was not intense. The software developers concentrated on doing software development and tested the piece of functionality they were assigned to verify whether it was working. Thereafter, it was assumed that the software was functional and ready for deployment, even without fully interrogating it. However, it is not easy for software developers to detect their own mistakes. So in many instances, the software testing team would only learn about the software when it was in production and already dysfunctional: "In our organisation they are able to bypass software testing, as long as they want the software to be implemented it gets implemented without being tested".

There was a change management as well as the release management team within the organisation. The change management team was responsible for evaluating the change requests logged, addressing issues relating to those changes and determining the impact of those changes on other projects. The release management team was responsible for approving changes and new software projects that needed to be implemented. Both teams, then, had processes and procedures that needed to be followed when dealing with changes or with new software projects needing implementation.

The other teams perceived the software testing team as a stumbling block or bottle neck regarding the deployment of changes and new software due to the release management process which required every single change or new software to be thoroughly tested and evaluated prior to deployment. Whenever the change or new software failed the software quality gates, it could not be approved by the release management. The strict release management process forced the software testing team to ensure that all changes and the new software was tested and evaluated before they could be implemented. One participant asserted that: "It was important to make sure that processes were followed because in that case software testing would not be bypassed".

According to the release management process, changes and software had to be rigorously tested in the Quality Assurance (QA) and Pre-Production (Pre-Prod) environments before they could be implemented, and they had to pass the necessary software quality gates. In one instance, to bypass this, one software tester spotted the business analyst changing the business requirement specification without consulting business: "I think the challenge that we are facing in our organisation is that processes are not followed". As a result, the other teams worked against the software testing team. The release management process involved the software testing team to make crucial decisions regarding the deployment of changes and new software. Therefore, the other teams perceived software testing as problematic because of this process. The relationship between software testers and software developers was challenging, especially when defects were logged against software developers. One software tester highlighted that: "When a software tester logs a defect against the software developer they would argue with you and even fight with you verbally".

Moments of translation: Interessement

The software development, testing and evaluation within the organisation attracted the attention of various actors. While some of these actors volunteered, others were obligated to partake in the testing and evaluation of software. From the management perspective, the actors included teams and individuals such as the change and release management teams, the business team, technology team, release manager, product owner, project managers, business analysts, software developers, software testers, and functional support personnel. These actors had various roles and responsibilities in the development, testing and evaluation of software: "We need to work together as a team including project managers, systems analysts, business analysts, test analysts and software developers". The actors had diverse interests in the testing and evaluation of software within the organisation, interests that were influenced by various factors. From the management perspective, they were expected to ensure that software produced enabled other government departments to perform their daily duties and render services to South African citizens. Some of those government departments included Environmental Affairs, the Development Bank of South Africa and Home Affairs. Each team

from IT and the business department would have a representative in the change management board: "We would have a representative from the department who would represent the department that requested the software or change".

The government departments are not there to make profit but to serve the citizens of the country. Therefore, it was important for those departments to have quality software that would enable them to fulfil their duties as public servants. At the operational level, business analysts, software developers, software testers, functional support and business end users were liable to deliver quality software. Some of the factors that influenced their interest included enforcing processes and standards for conducting software testing and evaluation, involvement of all project stakeholders at the early stages of developing the software and promoting the culture of collective teamwork within the organisation*:* "Actually what I believe needs to be done is to involve the software testers from the initiation of the project so that the tester can also give inputs and recommendations when coming to the testing of the software".

Some employees felt that it was human nature to bypass processes. Others felt that things could be done more quickly but the established processes prolonged things. So, even when processes were put in place, this did not necessarily mean that people would follow them. As a result, it was management responsibility to ensure that processes were followed. Failure in enforcing employees to follow processes would lead to the public company failing to produce quality software. Some participants stated that: "Processes are not followed. It is important to make sure that processes are followed because in that case testing will not be bypassed. "Processes are not followed at all and the management is doing nothing about the situation".

Other employees wanted to be part of the software testing team that provided other government departments with quality software. One business analyst even left the business analysis team for software testing, desiring to learn to use software testing tools that were adopted within the organisation. Those tools included IBM Rational tools and JMeter, open-source software used for performance testing. The public company was lacking in test automatons; therefore, they focused more on manual and performance testing: "The reason why automation was not viable technology for us to improve on our testing capability the maturity of our testing team is not at a mature level where we are ready to automate".

As a result, individuals needed to be empowered in order to automate software testing. Not all actors who were interested enrolled, though, so the enrolment of individuals was not completely successful as some people were not willing to take part in the testing of software and evaluation.

Moments of Translation: Enrolment

Participation of actors was pursued through different means within public company. The managers of various teams engaged with subordinates and negotiated their participation in the testing and evaluation of software. The team meetings were used as the negotiation platform to get those interested to enrol in the network that would test the requested software. Hence, software testing couldn't exist in isolation, various teams including project management, business analysts, software developers, software testing as well as business department collaboration of skills played a vital role. The managers of these teams used the power they had to encourage individuals to participate, to accept the roles and responsibilities allocated during the testing and evaluation of software: "We have got a test manager who overlooks the testing capability who monitors the projects and plan and coordinate the testing activities and who work closely to the test lead".

The change and release management team existed within thepublic company. Any new software requirement or enhancement to existing software had to undergo the rigours of this team. Even after the software had been developed or enhancements made, these needed to be thoroughly tested and evaluated because this team relied on the test results from the software testing team. But even though this process existed, some employees still went ahead and bypassed processes and standards. The same participant highlighted that: "You find out that the software was implemented without testing and we tested it while the software was in production those are the challenges we are facing".

The software testing team was faced with incessant challenges during the testing and evaluation of software. Some software developers did not have a good working relationship with software testers. Whenever software testers detected defects and logged them against respective software developer, arguments arose. As a result, software developers would think that software testers were attacking them or did not value the work they had produced. Meanwhile, project managers were committing to timelines without involving test managers. As the software testers were not always involved early in software development life cycle, teams involved in the software testing and evaluation were not treated the same. One software tester complained that: "Another challenge is that all the projects that I worked on the development time could be extended but system testing time is never extended, and I don't understand why".

The software testing team was treated unfairly because they had to sacrifice family time to work overtime to complete tasks assigned to them. Even so, they were perceived as enemies within the organisation. Even though they complained about how they were treated, management did not respond to their complaints. The public company adopted V-Model to develop software and enhance existing software. With the V-Model, each software development stage is countered by the software testing stage. After the delineation of requirements, the software testing team has to validate those requirements to create test requirements and test cases. Any ambiguous requirements needed to be clarified. The organisation also had software testing tools to capture all the testing activities.

Moments of Translation: Mobilisation

At this stage, employees acted on behalf of the organisation during the testing and evaluation of software in the organisation. Others were delegated to act as spokespersons to convince and persuade employees to participate in the delivery of quality software through its testing and evaluation. According to the organisational structure, the test manager was responsible for heading the testing team. Similarly, the change and release manager headed the change management board. The other members of various teams – such as project management, software development, and business analysts – reported to their respective managers. Mobilisations of employees were carried out along this structure. One participant explained as follows: "We have got a test manager who overlooks the testing capability who monitors the projects and plan and coordinate the testing activities and who work closely to the test lead". The test manager appointed team leads who were responsible for individual projects. These team leads, assigned projects and three to four software testers depending on the scope of the project, were responsible for supervising the software testers assigned and also reported on the testing status of the project. Additionally, the team leads compiled their reports concerning their projects and submitted them to the test manager: "Then they will do the execution and manage the day-to-day statuses and they will give the test manager a report on a daily basis".

The test manager would then gather all the reports from the team leads and consolidate them into one report which was then issued to the Head of Department. This process simplified reporting. The management was willing to accept the task of mobilisation which was linked to their performance appraisals. Mobilisation was a success and employees were enthusiastic about their contribution in the testing and evaluation of software. Once the software was developed and tested it could not be left hanging without being diffused to the target market for proper utilisation. This diffusion was made possible through the innovation decision process.

Innovation Decision Process: Knowledge

Unlike business users who used one or two types of software, such as SAP or Oracle, to conduct their day-to-day activities, the software testing team had to test a variety of software. To this end, it was imperative for the software testing team to acquire knowledge about each and every software needing testing and evaluating. During the testing and evaluation of SAP, for example, there were no resources internally who understood SAP within the organisation other than vendors. Therefore, this lack of knowledge regarding SAP rendered it difficult for the software testing team to efficiently test and evaluate the software: "I think software testers did not understand the functionality or know how to use SAP functioned which made software testing inefficient".

Vendors do not usually share knowledge about their work because this is how they generate money for their organisations or themselves. They intentionally keep certain knowledge to themselves so that they could be called in when something was not working. Even though that was the case, the test manager believed in upskilling his team members. The software tester's responsibility was only to execute the test cases, log defects and re-test those defects. The test analyst, who was senior to the software tester, was responsible for extracting test requirements from the business requirement specification, creating test cases and capturing them on Rational Quality Manager. They could also perform the functions of the software tester. Therefore, at times the test manager would assign the software tester the functions of the test analyst: "In my team I encourage cross skilling, where testers will fulfil the role of analysis so that they can grow towards becoming full fleshed test analyst but primarily we have got the test manager, test lead, test analyst and a tester".

The public company equipped the software testing team by sending them for training regarding the software testing tools the organisation had purchased. Not only that, but the company also took its software testers on international software testing courses to learn the concepts, various types of software testing and how to apply those types of testing. This training greatly improved the skills of software testers and helped them perform testing efficiently: "We attended International Software Testing Qualifications Board (ISTQB) which provides various testing courses, training on software testing and certification exams on those testing courses". The software testing team acquired knowledge through testing various software within the organisation. As a result, their experiences were amassed through their skills.

Innovation Decision Process: Persuasion

The IT department was persuaded by the change and release management to deliver quality software. All the various team members who were involved in the testing and evaluation of the software had to fulfil their roles to meet the timelines agreed upon with the particular government department requesting the software. It was not only the responsibility of the software testing team to deliver on time, but the entire project team. Software testing and evaluation was a collaborated duty that relied on other skills for timely fulfilment: "The requirements were gathered, software development took place, then software testing was performed then the software goes through the change management".

The business analysis team compiled the business requirement specification which was used to develop the software or make enhancements to the existing software. With these requirements, the software development team and software testing teams were able to perform their functions. As a result, this new innovation (software) persuaded the teams involved in the testing and evaluation of software to deliver quality software. However, there were challenges that occurred during the testing of the software: "Not enough time was allocated to testing the project and we were pushed to complete our testing within a short space of time".

That, however, did not discourage the software testing team from doing their work. The software testers worked tirelessly in order to deliver quality software. They worked extra hours during the week and even came in on weekends to work. They never sat

back and complained but worked as hard as possible because they understood that those who requested the software were expecting a finished product functioning as expected. One participant stated that: "It was crucial that when you promise to deliver a product within the specified time let the delivery occur within the agreed time".

The change and release management was expecting the testing results from the testing team to make informed decisions about the tested and evaluated software.

Innovation Decision Process: Decision

During software testing and evaluation, the software testing team decided to conduct various types of testing in order to deliver quality software. Once the team was satisfied with the quality of the software, they submitted the testing results to the change and release management. Based on the test results, the government department requesting the software was invited to conduct user acceptance testing. That was the final type of testing conducted with the customer (government department) with the help of the software testing team. The customer needed to test the software to ensure that it functioned as expected. When satisfied, the department needed to sign off the software to confirm their satisfaction with the delivered software: "The software testing team was involved in the user acceptance testing, we are the ones who facilitate it, we help the user with their acceptance criteria, the sign off".

The test results from the testing team were then issued to the change and release management team. The change and release management team used these test results to decide whether or not the software was ready for implementation. However, the change and release manager did not make all the decision alone; these were collaborative decisions with the managers of various teams within the IT department. One participant stated that: "We formed a change management board and we signed off the change collectively from a business owner, software testing, and software development perspective so it was being cleared by the release manager who assumed a role of a chairperson in the change management board".

Innovation Decision Process: Implementation

At this stage, the functional support team worked closely with the change and release management team as well as the software development team to implement the tested software at public company. The functional support team needed the implementation plan created by the software development as well as the packaged software to deploy the software to production. The functional support team also had to configure the production environment prior to the deployment of software. The change management process still continued: "If there was a change in the software, the change request was logged, and it followed all the necessary channels like it was developed, tested by software developers, tested by software testers and tested by functional specialist then it was deployed to production".

Once the software was deployed, the change management team altered the status of the implemented software to 'complete'. The IT department at public company officially handed over the software to the government department that requested the software. The product owner of the software notified their staff about the new software prepared for use. As expected, some employees were delighted to use the software and to even read more about the usefulness of the software, while others were sceptical, too comfortable with how they did things prior to the new software deployment. One software tester asserted that: "Our organisation produced quality software for development bank of South Africa and users are happy to use the system".

The manager who requested the software for their department had to engage the power bestowed with the position to encourage staff to use the new software. Staff needed to understand that the software intended to simplify their daily activities as well as to render efficient services to the citizens of the country. The staff also needed to understand that the organisation spent time and money in developing the software to ensure the government department was efficient in rendering these services to the citizens of South Africa. Therefore, the organisation had managers who were willing to engage their staff, especially when issues and conflicts arose.

Innovation Decision Process: Confirmation

At this stage, it was necessary to recognise the benefits of using the innovation (new software), making the software part of the ongoing routine and promoting it to others within the department. As a result, this improves the government department's opportunity to function effectively. The public servants who were end users were able to perform their daily duties with the new software as well as render the service to the citizens of the country. One participant highlighted that: "Our organisation produced quality software for development bank of South Africa and users are happy to use the system".

The software is still being used to render services to the public. The software developed at the public company enabled the government department to meet its mandate of delivering quality service to the citizens of the country. The end user output indicated that the government department made the right decision by implementing this new software. The government was heading in the right direction by introducing this new software, increasing efficiency. One participant praised the software testing: "I am not saying the software testing team is the best of the best because even them they are in a growing phase, but I think it is a step in the right direction".

The government department finalised the decision to continue using the software (innovation), a stage of confirmation that the government department has made the right decision.

Findings and Discussion

Based on the analysis of the empirical data from the public company, six factors were found to be critical to building a decision support system for testing and evaluating software in the company.

As shown in Figure 1, the factors include software evaluation, process oriented, implementation policy, change management, power relationship and organisational structure. These factors are presented in Figure 1 below:

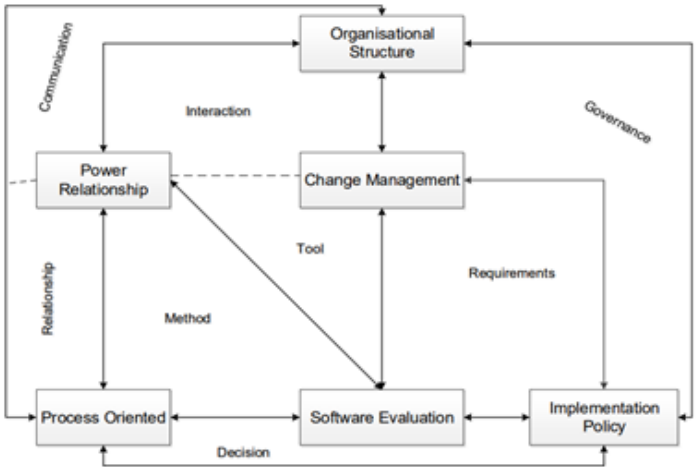

Figure 1: Factors influencing software testing and evaluation.

Software evaluation

At public company, software was evaluated from various perspectives such as sustainability, maintainability, usability and functionality. The focus on software sustainability was to evaluate whether the software can continue to be available for future purposes, such as its compatibility with newer platforms and changing business and technical requirements. The emphasis was on continuity mainly because some software gets discontinued after a short period of time. As a consequence, it become a loss to the organisation from a return on investment (ROI) viewpoint. Also, the software was tested for maintainability in order to ensure the ease in which the software can be enhanced for additional requirements, enhancement to correct faults detected, and adaptability to change of environments to improve performance. Organisations want to have flexible software that could be modified as change happens within the organisation.

Therefore, all software, new or with enhancements, needs to be thoroughly tested and evaluated to ensure it fulfils the requirements. This is an action that requires decisions at various levels, including business (product owners), management (product sponsor) and IT (technical experts). At public company, a committee was responsible for accepting requests to develop or modify software for government 147 departments. Once the development or modifications were completed, the software testing team had to thoroughly test and evaluate the software, employing various testing methods and tools to perform the testing. Through those methods and tools, test results were produced to report on the software testing status. Other teams, such as the change and release management team, relied on those test results to make informed decisions regarding implementation of the software.

These test results included the test cycles performed during testing, the total number of test cases executed, and the total number of defects detected and resolved. Moreover, the test results included recommendations from the software testing team about the software tested and evaluated. That information assisted the change management team to make appropriate decisions as the test result determined whether or not the software was ready for deployment. The change management adopted implementation policy to deploy the software. However, there were situations where the change management was bypassed, which could have been associated to power relationships that existed. This often-caused software failure, which resulted in immediate damage control that the organisation was always not prepared to carry out.

Once the software has been deployed, the management used their power to encourage end users to use the software. It was not obvious that when the software was deployed within the organisation, end users would automatically adopt it. Hence, the management had to use the power inherent in their positions to diffuse the new software or the enhancement to the government department that requested the software. It was critical that the software function as expected as the potential of dysfunctional software provided a reason for end users to reject the software. The process of testing and evaluating software was quite intense.

Process oriented

As empirically revealed, software testing is process-oriented in that it follows a set of steps, instructions, guidelines and policies to complete, with the intention of producing quality software. Without such intensive and rigorous processes, quality would be difficult to achieve. The testing of software at public company was not always automated; manual processes were also involved. Software testers had to execute a set of test cases manually to ensure that the software functioned as expected. Manual testing was a tedious process, repetitive and requiring full concentration of the tester. One of the challenges was that, at times, software testers were not allocated enough time to properly carry out their tasks. This resulted in software testers working long hours, occasionally causing fatigue, which then detrimentally affected the quality of their testing – a poor cycle. Therefore, some test cases might mistakenly pass or fail. However, software testing automation is required to fast track the testing process by using the necessary tools. Currently, the public company make use of manual tools to capture test requirements, test cases and defects (IBM Rational Tools) as well as the performance testing tool (JMeter). These tools and processes require a decision support system that can seamlessly enable and support it.

The process includes the following: defects detected during the testing and evaluation of software which were logged and needed to be fixed by software developers. However, at times these defects erupted conflicts between the software developer and software tester who logged the defect. Such conflicts manifest from the relation that they have, a relationship of power. For example, the software developer would inform the software tester that the defect logged was not a defect. These conflicts tarnished the relationship

between the two parties, who saw themselves as networks with the implication being that the quality of software was worsened.

Software testers needed to be analytical, innovative and able to communicate appropriately. Analytical skills enabled them to understand both the business requirements specification and the other group (network) of people that contribute to complete the tasks of software so that it can be implemented and used (diffused). The technical experts (software testers and developers) need to be more innovative to improve on the issues and factors that were not explicitly stated on the business requirement specification. This can be achieved through a decision support system that can validate various steps and required actions.

Communication also played a vital role because software testers needed to be able communicate well with software developers in terms of building a relationship which can bridge an understanding of defects. Failure in understanding the business requirements meant that software testers would have neglected other scenarios unintentionally and the software would only be partially tested, potentially hindering the quality of the product. Software testers also needed to understand the environment in which the software was deployed. Software fails not only due to functional requirements but can fail due to non-functional requirements.

Implementation Policy

Software testing and evaluation are intensive types of processes, which many employees in the organisation struggled to comprehend and abide by. Thus, an implementation policy is required, which was not in existence at public company as revealed from the empirical evidence. Many of the employees recognise the significance of such policy. The management buy-in was necessary to enforce the necessary measures in implementing policy for software testing and evaluation. The software should be tested and evaluated through this intensive process for quality purposes. Necessary software testing processes must be followed as bypassing these processes affected the quality of the software.

It was necessary for the management to offer their support in terms of enforcing software testing processes between the project stakeholders. The software testing team has to abide by processes when testing software. Both the management and technical teams have to ensure the implementation policies are followed in the processes of development, testing and evaluation of software in the organisation. This can be done through governance, forming part of a system for decision support of the entire software production cycle.

This approach can enable requirements to be fulfilled, and as well prevent software from being deployed without the knowledge of testing and evaluation criteria and processes, which impact how change is currently managed in the organisation. For example, there were instances whereby software testing was bypassed, and yet the software still found its way to production. As a result, the software failed and the same team that was earlier bypassed was then requested to assist with the testing of that software. The rationale for not observing the implementation policies leads to bad quality software being implemented in production.

Change Management

Change management can be linked back to the implementation policy. Any new software or enhancements to existing software were managed through the change management process. This was to ensure that the software functions as expected and meets the business requirements. Proper governance needed to be enforced by the change management team to ensure that only quality software was deployed to production within the organisation. Change management rely on the test results that are provided by the software testing team to make informed decision about implementing the software.

The change management can be manual or automated, but through a system that supports the entire process of software testing and evaluation. The change management includes completion and signed off documentation such as specification, change and closure reports. Other signed off documents, such as test plan and closure reports, serve as the entry criteria to the change management team stating the outcome of software testing. As a result, decisions are influenced and guided by the test results. Thereafter, the implementation policy can be adopted (or diffused) to allow the enhancement or new software to be deployed to production. This process assured the management that the software deployed can be of high quality and would not have a negative impact on existing software within the government department that requested for it. Bypassing the necessary processes can lead to implementation of poor-quality software in the organisation.

Power Relationship

In the software development project, various stakeholders, including project managers, business analysts, software developers, software testers and functional support personnel were involved at the public company. In the process of development, testing and evaluation of software the stakeholders interacted, through which relationships were created both consciously and unconsciously. The conscious relationships were often created through organisational structure. For example, software developers physically communicate to resolve logged defects. Unconscious relationships were guided by informal friendships and favouritisms (e.g. some software testers communicated with those with whom they were comfortable rather than what the structure of the organisation dictated in the deployment of software into production).

However, relationships could be twofold, good or bad, and often used as source of power. Good relationships are earned through respect, effective communication, and delivering tasks assigned to project stakeholders on time. This kind of relationship needs to be maintained as it motivates project stakeholders to perform their tasks to the best of their ability. As a result, quality software could be delivered before or on time. Good relationships also enable the project stakeholders to abide by software testing processes

and policies to deliver quality software. Process-oriented implementation policy as well as change management played a vital role in managing power relationships, in that organisation objectives took precedence over individual preferences. When power relationships are managed well, it leads to conducive communication within an environment which enacts improved productivity of software delivery.

The management of power relationships can be implemented through an automated system that can support the entire testing and evaluation of software within an organisation. Otherwise, power relationships can also lead to bullying within workplace. As revealed from the analysis, some software developers bullied software testers over the defects they detected in the software. For instance, the software developer informing the software tester that they are unable to develop the software, the only thing they knew was to test it, whereas the software developer can develop and test. That resulted in a bad relationship that came about through disrespect and undermining roles of other project stakeholders. Such a relationship rendered the team dysfunctional and poor-quality software was produced. This impacted the testing results negatively.

Organisational Structure

The organisational structure is a hierarchical arrangement in terms of authority, communications, rights and duties of employees in an organisation. Some of the essentials of the organisational structure include the following, (1) power of each or group of individuals is controlled through formal communication along various levels in the organisation; (2) the activities of change management are governed within the organisation's aim and objectives; and (3) policies are implemented by using the governance. These essential factors can be enabled and supported through a decision support system to reduce complexity and conflict of interest.

The organisational structure therefore helps in many ways such as, to determine how roles, power and responsibilities are assigned; how roles and responsibilities are controlled and coordinated; and how information flows between the different levels of management in the testing, evaluation and deployment of software. Every employee within the organisation when employed was assigned roles and responsibilities, functions they were expected to perform. Such roles and responsibilities come with some sort of power, to manage subordinates and control activities during testing and evaluation of software. For example, the test manager was responsible for creating a test strategy, test plan and various reports. They also supervised software testers to ensure that they were doing what they were tasked to do and to deliver on time. The test manager was expected by superiors to report regularly on the testing status whenever required. Therefore, the test manager had the power to assign tasks to software testers at any given time and discipline them if they were not fulfilling those tasks. Also, the responsibility of the test manager was higher than of a tester.

When project stakeholders undermined processes due to power they hold, organisational structure could be the solution to that problem. The management needed to manage the situation so that the software could be tested and evaluated. The management needed to outline and make project stakeholders aware of the software testing processes and implementation policies, and then enforce them for testing the software. Unruly behaviour by some project stakeholders also needed to be minimised to ensure that such behaviour did not undermine and disrespect other team members. The organisational structure needed to align processes, policies, changes and power to ensure that the testing of software is performed accordingly. The project stakeholders needed to understand the effect that software testing could have when not performed. Defects would be detected in production by end users. As a result, IT department would appear as people who did not know what they were doing. As a ripple effect, management would be blamed for failing to do their work. Therefore, the organisational structure helped align processes in order for the project stakeholders to function properly.

Conclusion

Any product that is developed needs to be evaluated and tested robustly to ensure that it is of quality prior to reaching the public. The objective is to ensure that it serves the purpose of what it was designed for. Software within the company is developed to be used to perform its day-to-day operations. As a result, it enables the company to be productive, competitive and efficient in its business. ANT and DoI were used as a lens to analyze data that was collected within the public company. Six factors were found to be critical to improve how software is tested and evaluated within the company. Thus, software testing is the critical part of software development to ensure the quality and stability of the software developed.

References

1. Ahamed SSR. 2009. Study the feasibility and importance of software testing: An Analysis.
2. Williams L. White-Box Testing. 2026. https://students.cs.byu.edu/~cs340ta/spring2018/readings/WhiteBox.pdf.
3. Irena J. Software Testing Methods and Techniques. The IPSI BgD Transactions on Internet Research. 2008; 30-41.
4. Hussain T, Singh S. A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing. IJAPRR. 2015; 2: 1-8.
5. Mishra S, Pradhan A. Software Testing Techniques Adopted in Corporate Sectors: A Precise Study. International Journal of Emerging Technology and Advanced Engineering. 2012; 2: 290-295.
6. Khan ME, Khan F. A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. IJACSA. 2012; 3: 12-15.
7. Aichernig BK. Systematic Black-Box Testing of Computer-Based Systems through Formal Abstraction Techniques. PhD thesis, Institute for Software Technology, TU Graz, Austria, Supervisor: Peter Lucas (January 2001)
8. Khan ME. Different Approaches to White Box Testing Technique for Finding Errors. IJSEA. 2011; 5: 1-14.

9. Nidhra S, Dondeti J. Black box and white box techniques - A literature review. IJESA. 2012; 2: 29-50.
10. Steegmans E, Bekaert P, Devos F, et al. Black & White Testing: Bridging Black Box Testing and White Box Testing. Conferentie Software Testing: Beheers Optimaal de Risicos van IT in uw Business. 2004; 1-12.
11. Acharya S, Pandya V. Bridge between Black Box and White Box – Grey Box.
12. Sawant AA, Bari PH, Chawan PM. Software Testing Techniques and Strategies. IJERA. 2012; 1: 980-986.
13. Saxena R, Singh M. Grey Box Testing: Proactive Methodology for the Future Design of Test Cases to Reduce Overall System Cost. Journal of Basic and Applied Engineering Research. 2014; 1: 62-66.
14. Bhasin A, Kumar M. Study of White Box, Black Box and Grey Box Testing Techniques. International Journal of Research in Engineering Advanced Technology. 2015; 3: 23-27.
15. Kothari CR. Research Methodology Methods and Techniques. 2nd ed. New Age International Publishers. 2004.
16. Reiter B. The Epistemology and Methodology of Exploratory Social Science Research: Crossing Popper with Marcuse. Government and International Affairs Faculty Publications. 2013; 1-17.
17. Lihosit J. Breaking Down the Black Box: How Actor Network Theory Can Help Librarians Better Train Law Students in Legal Research Techniques. Law Library Journal. 2014; 106: 211-220.
18. Alqahtani S, Wamba SF. Determinants of RFID Technology Adoption Intention in the Saudi Retail Industry: An Empirical Study. 45th Hawaii International Conference on System Sciences. 2012; 4720-4729.
19. Montfort D, Brown S, Pegg JM. An Investigation of the Adoption of an Assessment Instrument for Capstone Design Courses. Proceedings of the 39th IEEE international conference on Frontiers in education conference. 2009; 148-153.