AN AGILE INFORMATION FLOW CONSOLIDATOR FOR DELIVERY OF QUALITY SOFTWARE PROJECTS: TECHNOLOGICAL PERSPECTIVE FROM A SOUTH AFRICAN START-UP

By

ABDEL KADER DOUKOURE GAOUSSOU

Submitted in accordance with the requirements

For the degree of

DOCTOR OF PHILOSOPHY

In the subject

INFORMATION SYSTEMS

At the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: Professor Ernest Mnkandla

DATE SUBMITTED: 15 MAY 2019

THESIS SUMMARY

In today’s knowledge-based economy, modern organisations understand the importance of technology in their quest to be considered global leaders. South African markets like others worldwide are regularly flooded with the latest technology trends which can complicate the acquisition, use, management and maintenance of software. To achieve a competitive edge, companies tend to leverage agile methods with the best possible combination of innovative supporting tools as a key differentiator. Software technology firms are in this light faced with determining how to leverage technology and efficient development processes for them to consistently deliver quality software projects and solutions to their customer base.

Previous studies have discussed the importance of software development processes from a project management perspective. African academia has immensely contributed in terms of software development and project management research which has focused on modern frameworks, methodologies as well as project management techniques. While the current research continues with this tradition by presenting the pertinence of modern agile methodologies, it additionally further describes modern agile development processes tailored in a sub-Saharan context. The study also aims novelty by showing how innovative sometimes disruptive technology tools can contribute to producing African software solutions to African problems. To this end, the thesis contains an experimental case study where a web portal is prototyped to assist firms with the management of agile project management and engineering related activities.

Literature review, semi-structure interviews as well as direct observations from the industry use case are used as data sources. Underpinned by an Activity Theory analytical framework, the qualitative data is analysed by leveraging content and thematic oriented techniques.

This study aims to contribute to software engineering as well as the information systems body of knowledge in general. The research hence ambitions to propose a practical framework to promote the delivery of quality software projects and products.

For this thesis, such a framework was designed around an information system which helps organizations better manage agile project management and engineering related activities.

Key terms:

Quality; SME; Agile; Cloud Computing; DevOps; Kanban; Jenkins; Scrum; Activity Theory.

DECLARATION

Name:	Mr Abdel Kader Doukoure Gaoussou
Student number:	36660213
Degree:	PhD in Information Systems

Exact wording of the title of the thesis as appearing on the electronic copy submitted for examination:

An agile information flow consolidator for delivery of quality software projects: Technological perspective from a South African start-up

---

I declare that the above thesis is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

I further declare that I submitted the thesis to originality checking software and that it falls within the accepted requirements for originality.

I further declare that I have not previously submitted this work, or part of it, for examination at Unisa for another qualification or at any other higher education institution.

(The thesis will not be examined unless this statement has been submitted.)

[illegible] ___12-05-2019________

SIGNATURE DATE

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

- **AT** – Activity Theory
- **BPM** – Business Process Model
- **BPMN** – Business Process Model Notation
- **CD** – Continuous Delivery
- **CI** – Continuous Integration
- **CMMI** – Capability Maturity Model Integration
- **DevOps** – Development and Operations
- **GZAPMC** – GZ Agile Project Management Consolidator
- **GZCS** – GZ Consulting Services
- **ICT** – Information Communication Technology
- **IIDD** – Iterative and Incremental Design and Development
- **IS** – Information System
- **ISO** – International Organization for Standardization
- **LOC** – Lines of Code
- **ME** – Method Engineering
- **SCM** – Source Code Management
- **SFSC** – Software Focused Supply Chain
- **SEI** – Software Engineering Institute
- **SME** – Small and Medium-sized Enterprises
- **SMME** – Small Medium and Micro-sized Enterprises
- **SPEM**– System Process Engineering Meta-Model
- **SPI** – Software Process Improvement
- **SQA** – Software Quality Insurance
- **TOGAF** – The Open Group Architecture Framework
- **XP** – Extreme Programming
- **UX** – User Experience

DEDICATION

To Maimouna and Ousmane Doukoure.

ACKNOWLEDGMENTS

I am grateful to the Almighty for all blessings which have afforded me this opportunity.

I also must eternally thank my mentor and supervisor, Professor Ernest Mnkandla. This thesis would not have been completed without his invaluable guidance, generosity and wisdom. Our institutions and academic community are blessed to have intellectuals such as yourself for my fellow students to look up to.

Finally, I would like to thank my family, loved ones and close friends for their encouragements, love and support throughout this journey.

PUBLICATIONS FROM THE THESIS

Gaoussou, AKD. & Mnkandla, E. (2018). Facilitating the Management of Agile and DevOps Activities: Implementation of a Data Consolidator. In *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)* (pp. 1-6). IEEE.

Gaoussou, AKD. & Mnkandla, E. (2018). Managing Agile and Dev-Ops Activities through a Consolidated Web Portal: South African Case Study. In *11th IADIS International Conference on Information Systems (IS 2018),* Lisbon, Portugal, pp. 121-128.

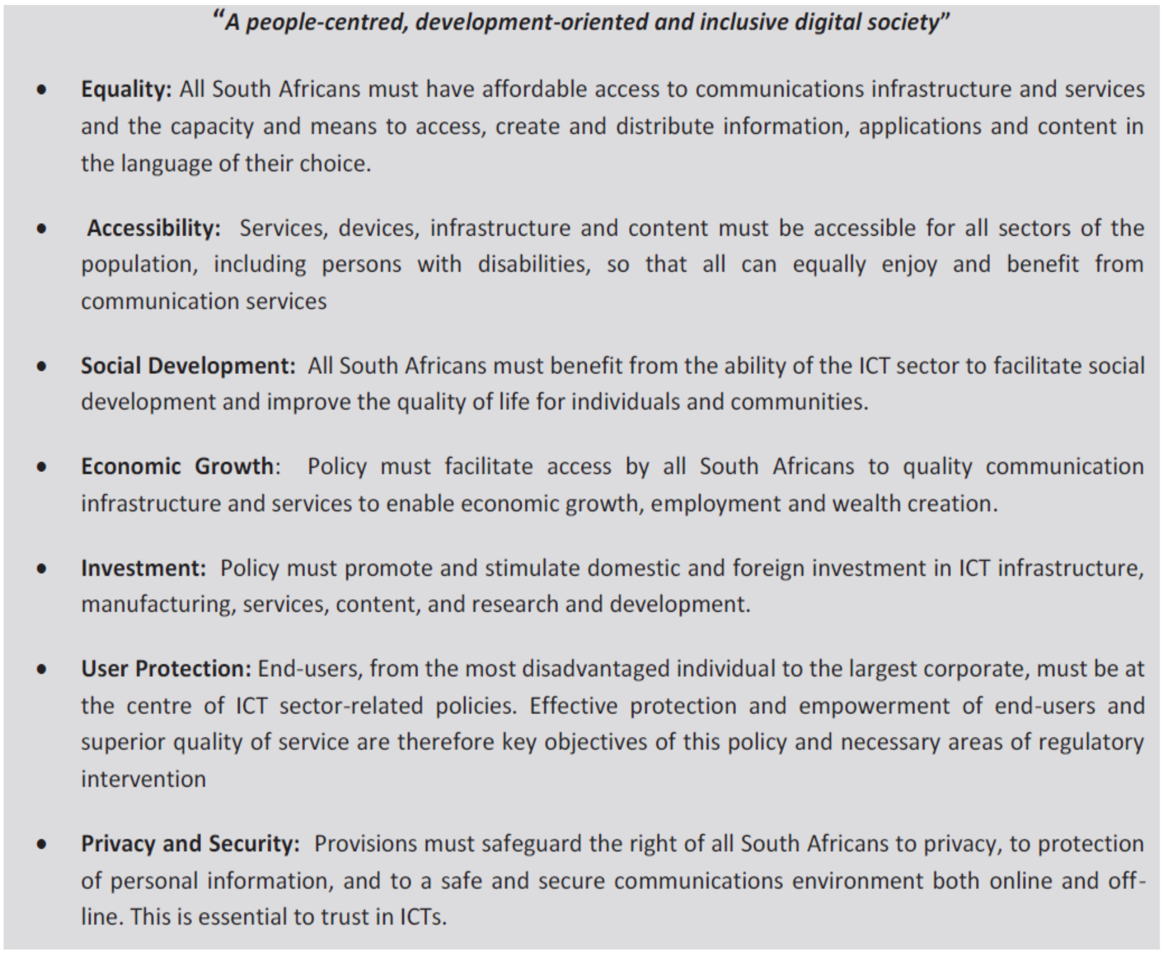
1. CHAPTER ONE: RESEARCH CONTEXT

1.1 Introduction

In order to catch up with industrially advanced nations and maintain sustainable growth, emerging economies are imitating and constantly adapting technologies streaming in from the developed world (UNIDO 2016, p. 5). This presents challenges for organisations as remaining competitive in today's knowledge-based economy is increasingly considered a question of survival. In order to stay ahead, many companies tend to leverage Information communication Technologies (ICTs) as a key differentiator (Baller, Dutta & Lanvin 2016, p. 8).

The worldwide digitalization phenomenon 's impact on societies are commonly identified as a Fourth Industrial Revolution (IODSA 2016, p. 19). Information Communication Technologies (ICTs) powered by software systems form an essential part of this profound transformation of modern societies and economies.

ICTs can be described as the group of digital technologies which facilitate communications as well as the storage, aggregation and distribution of information throughout organisations (Ritchie & Brindley 2005, p. 20). Policy work compiled by the South African National Planning Commission predicts that at the horizon of 2030, ICTs will be indispensable to achieve an inclusive economy (National Planning Commission 2010, p. 190). This analysis is in line with South Africa's government vision which considers technology as a crucial medium in its quest to achieve effective socio-economic improvement of previously disadvantaged communities. Cwele (2016) establishes the objectives which need to be met for cabinet's ICT policy to be successful:


"A people-centred, development-oriented and inclusive digital society"

- **Equality:** All South Africans must have affordable access to communications infrastructure and services and the capacity and means to access, create and distribute information, applications and content in the language of their choice.
- **Accessibility:** Services, devices, infrastructure and content must be accessible for all sectors of the population, including persons with disabilities, so that all can equally enjoy and benefit from communication services
- **Social Development:** All South Africans must benefit from the ability of the ICT sector to facilitate social development and improve the quality of life for individuals and communities.
- **Economic Growth**: Policy must facilitate access by all South Africans to quality communication infrastructure and services to enable economic growth, employment and wealth creation.
- **Investment:** Policy must promote and stimulate domestic and foreign investment in ICT infrastructure, manufacturing, services, content, and research and development.
- **User Protection:** End-users, from the most disadvantaged individual to the largest corporate, must be at the centre of ICT sector-related policies. Effective protection and empowerment of end-users and superior quality of service are therefore key objectives of this policy and necessary areas of regulatory intervention
- **Privacy and Security:** Provisions must safeguard the right of all South Africans to privacy, to protection of personal information, and to a safe and secure communications environment both online and off-line. This is essential to trust in ICTs.

Figure 1.1: ICT Policy White Paper Objectives (Cwele 2016, p. 11)

While government's progressive vision in terms of the role of ICTs in society and the economy is undeniable, South African markets like others worldwide are regularly flooded with the latest technology trends. This can complicate the acquisition, use, management and maintenance of software. Nonetheless, the local industry has historically presented a certain appeal to national and international players. The local software engineering sector can be illustrated as an example to describe this situation.

As far back as 2010, a Gartner press release, had identified South Africa as a top tier software engineering outsourcing alternative (Gartner 2010, para. 4). Five years later, the country's attractiveness in terms of privileged outsource destinations for multi-nationals remained notwithstanding expected macro-economic difficulties. In 2015, an assessment by International Data Corporation (IDC) described how despite unemployment challenges, a difficult economic outlook, unstable currency fluctuations, as well as energy supply shortages; ICT spend in South Africa was still expected to grow up by 2.6% the following year (IDC 2015, para. 6). The same year, Gartner predicted an increase in IT spending up 5.1 % to reach $ 26.6 billion in 2016

(Gartner 2015, para. 1). Enterprise Software spending for example was expected to grow to 9.3% up in 2016 (Gartner 2015, fig. 1). The same Gartner report additionally explained that as mobile telephony spending was expected to reach more than $5 billion in 2018, South African engineers were well placed to profit from this growth (Gartner 2015, para. 7). This led Gartner's to posit that local firms understand the intricacies of conduction business in Africa (Gartner 2015, para. 8). It was hence observed that the engineering skills and innovation required to solve local problems can be found in abundance on the continent (ibid., para. 8). In the same vein of analysis, workforce current talent trends seem to suggest that the South African market is becoming increasingly competitive (PWC 2016a, para. 14). At a macro level, the cut throat nature of liberal markets presents technology firms with an opportunity to gain a competitive edge. Software project teams in particular can thrive by delivering quality software projects and solutions to the organization and its customers. In this light, local software engineering organisations like others worldwide are faced with the problems of quality. In practice, quality presents specific challenges inherent to software engineering. This is largely due to intrinsic notional aspects of software at its core.

1.2 Background to the problem

Software engineering typically entails developing software in a systematic, scientific and quantifiable manner which also involves applying the same discipline to the operations and maintenance aspect of software projects (ISO/IEC & IEEE 2010, p. 331). The application of sound software engineering principles and practices can help organisations improve their development processes while mitigating complexities inherent in non-trivial software solution designs and implementations (Kalermo & Rissanen 2002, p. 14).

The emergence of software products has disrupted most areas of our social economic modern lives; as such, its pervasiveness plays an essential part in almost all consumer goods and services available to customers (Deek, McHugh & Eljabiri 2005, p. 13). Studies have for instance highlighted that even in traditional retail, software is sometimes the value added differentiator which distinguishes competitors and can determine market share between competitors (Tan & Yuan 2005, p. 187).

Despite the availability of well-defined practical approaches to planning and executing engineering projects, software itself can be described as largely intangible. This can be explained in part by the fact that one cannot physically feel code content. Additionally, a software product is not simply visualised by non-professionals. To describe in a more detailed manner this nature of software, the following elements have to be taken into account (Lethbridge & Laganiere 2004, pp. 1–3) :

- *After production, software pieces are typically easily duplicated, or reproduced.* Costs incurred amid most software development projects typically appear during development and not manufacturing. The main issue is that while software is expensive in terms of financial costs and intellectual capital to build, once it is built, it is relatively simple to reproduce, duplicate.
- *The software industry is labour intensive.* In most manufacturing industries for instance, the intensive use of machinery has helped these sectors produce more with less toil. This is not truly applicable in the software industry as to really automate software design and engineering, one would require truly "intelligent" or "sentient" machines. Despite many advances in artificial intelligence, the software industry hasn't reached this level yet.
- *It is very easy for the most basic programmer to create functional software that is poorly designed.* In the software industry, developers can often write complex systems which fulfil their intended purpose despite being almost impossible to maintain or enhance. In other industries such as the electrical engineering or civil engineering domain, design flaws are much easier to detect, identify and correct early.
- *Modifying software can be a trivial task.* In complex projects however, making the appropriate changes is challenging. A developer who takes over code he or she did not initially write often makes changes without understanding all the subtleties of the software he or she is working on, because of this, fixing issues often introduces new bugs to the product.
- *Software integrity tends to deteriorate as its design evolves.* This point is linked with the previous one as code changes often result in changes with the initial algorithm of software products. Hence, over time, design changes in successive software versions may result in such poor performance that product development requires discontinuation and a complete overhaul of the design is the only option.

As mentioned in earlier sections, software systems are largely abstract and intangible; consequently, they can be very difficult to describe or explain to people outside the engineering field (Avison & Fitzgerald in Mwansa 2015, p. 27). This ungraspable feature of software's nature influenced development practices early on (ibid., 2015).

Originally, software applications were largely developed without any real consistent methodology until the 1960s (Fitzgerald 2012, p. 89). The direct consequence of this lack of rigorous scientific approach yielded many problems in the development and delivery of software (ibid., 2012). In effect, as early as the 1960s and 70s, organisations struggled with software project delays, failures, as well as customer increased concerns with demands of improved functionality and quality (Deek, McHugh & Eljabiri 2005, p. 177).

Considered as a whole, these factors which highlight the nature of software have contributed to what was described as a *"software crisis"*.

The software crisis was a term coined to describe the challenges of maintaining software and a need for new approaches to engineering it (Mnkandla 2009, pp. 1–3). In the early 90s for example, the situation raised flags for some professional bodies such as the Standish Group which considered software engineering high cancellation rates as highly problematic (Khaled & Gunes 2004, p. 84). Individual researchers suggested on the other hand that despite half a century of noteworthy advancements, software developers still lacked an effective and mature enough discipline to address growing requirements for a knowledge based economy (Gibbs 1994, p. 87).

Though this crisis was alleviated with structured development methodologies, the issue of consistent software quality delivery itself remains. As scholars and industry professionals attempt to establish a common set of measurements to gage the quality of software products, three parameters have historically been considered as consensual. These metrics could be classified as software process quality, product quality, as well as project quality itself (Yinfen 2011, p. 2699). Process and product quality are often associated because they have traditionally been understood as inter-dependent. Previous studies have for instance discussed how it is by having efficient quality processes, that organizations can output quality software products. In these contexts, processes refer to the quality development related activities. Software product quality on the other hand refers to the delivered artefact's overall quality and the satisfaction it provides to its intended target audience (Jain, Sharma and Ahuja, 2018, p. 813). Finally, project quality is an all-encompassing notion which has at its basis teams involved in ensuring projects are successfully delivered to stakeholders. Using projects as a measurable artefact to gage quality can hence be considered as a tactical instrument to improve quality delivery in software engineering contexts itself (Yinfen 2011, p. 2700). Ultimately, academia as well as industry stakeholders have a vested interest in understanding all elements which can enhance their ability to deliver quality to customers.

In practice, contemporary research is still concerned with quality in the software engineering world. Considering that the Agile Manifesto was issued more than a decade ago, academics are still conducting studies on the topic of quality in terms of software requirements specifications amongst others for instance (Medeiros et al. 2016, para.8).

It is hence against this contextual background that the current study aimed at exploring the development of an agile oriented framework revolving around quality software project delivery to markets. While this is argued to be enabled through the adoption of tailored agile software development processes, the current study also describes how these processes can be made more effective by the practical usage of modern, agile, adaptive and disruptive technological tools.

1.3 Problem statement

Despite a seemingly favourable macro-economic environment, the South African software industry is not exempt from concerns regarding the delivery of quality software projects and solutions. From a resource management perspective for instance, project managers are often forced to prioritize two options out of timely delivery, specifications adherence or budget constraints (Bot 2019, para. 1). Without a contextually adapted framework to modern software development, the usage or non-usage of innovative/agile approaches to software engineering will not resolve persistent quality concerns. Moreover, in certain organizations which claim to have solved software project risks and concerns by simply adopting agile methodologies, some have gone as far as using the agile approach as somewhat of an excuse and justification for having no or insignificant processes in place (Ambler & Lines 2012, p. 2).

It could hence be deemed relevant to develop a practical agile oriented framework supported by related technological tools for the continuous delivery of quality software projects and solutions to markets. This could arguably enhance both large as well as smaller organisations' ability to better adapt and react to the current knowledge-based economy in which markets require adaptability and quick reaction times from products and services provider (Bot 2019, para. 3).

1.4 Research questions

This thesis was underpinned by a central research question:

"How could an agile oriented framework be leveraged to facilitate delivery of quality software projects for organizations operating in sub-Saharan settings?"

This main research question required an exploration of concepts related to software engineering at large, traditional and current approaches to delivering software projects, as well as a grasp of the challenges and opportunities presented to large and smaller businesses ambitioning to shift towards agile models of operations. To further explicit issues inferred from this central interrogation, four sub-questions where formulated to address the research problem at hand:

- What does quality entail in software engineering projects and how does the modern software engineering industry deal with this issue?
- How can an organisation tailor its development processes in an agile manner and facilitate its delivery to market?
- What are the modern disruptive technological tools at the disposal of a South African small and medium enterprise (SME) to achieve the continuous delivery of quality software projects and products to market?
- How can an agile oriented information system complement these tools to facilitate an African SME's delivery of quality software projects and solutions to market?

1.5 Research Objectives

Derived from the questions above, this research sought to:

- **Establish a clear understanding of what quality software projects entail in modern software engineering organisations.** Literature review is used to arrive at a comprehensive and global understanding of the concept and related terms.
- **Describe the importance of tailoring development processes to organisational and market specific needs.** This objective is addressed through literature survey as well as direct observations of day to day activities of engineering and project management staff during an industry case study.
- **Present technological tools available to facilitate agile development and enable quality software project delivery.** This objective is implemented through experimentations with some agile and cloud technologies, browsing of online resources, academic research readings, as well as industry white papers exploration.

- **Develop an information system to complement existing tools, to facilitate the delivery of quality agile development projects and solutions.** For this objective, gaps identified through observations of agile methodology and related technology usage of GZ Consulting Services staff is used to develop an information system to complement existing tools.
- **Propose and deploy the developed tool at GZ Consulting Services.** For this objective, the prototype built is put at the disposition of management and engineering staff for practical usage during day to day activities.

1.6 Rationale for the research

Previous studies have discussed the importance of software engineering through a project management point of view. African academia has immensely contributed in terms of software development and project management research which has focused on modern software development frameworks, methodologies and project management techniques. While the current research continues with this tradition by presenting the pertinence of modern agile methodologies, it additionally further describes modern agile development processes tailored in a sub-Saharan context. The study also aims novelty by showing how innovative sometimes disruptive technology tools can contribute to producing African software solutions to African problems. This research suggests an approach to facilitate the delivery of quality products during engineering projects by aggregating data from various agile tools into one consolidated web portal to present a reliable unified view of information for project stakeholders.

1.7 Delimitations

Although literature reviewed during this study discusses quality regarding software engineering, an emphasis is placed on the process of building software in an agile manner to facilitate the continuous delivery of quality software to markets. Topics such as user requirements specifications quality for instance are not thoroughly explored. This is purposely done considering that one of the core Agile Manifesto values is that working software is more important than previous foundational blocks of traditional rigid software development models (Medeiros et al. 2016, para. 2). Nonetheless, structural implications for software quality delivery will be discussed in Chapter Two to understand how various dimensions of software quality tend to influence its perceived business value.

Another aspect to consider is that of the technological tools presented to facilitate this continuous delivery of quality software projects. These are purposely identified based on their low cost when no open-source alternative could be found. This was done as the reality of

African markets is that most software start-ups have difficulty finding capital locally. Additionally, the case study of this thesis is situated at a sub-Saharan technology SME based in Johannesburg, South Africa. The framework developed at the term of this study is hence not necessarily applicable to larger multinational technology corporations operating within different economic industrial contexts.

1.8 Ethics considerations

Research ethics policies are established by academic institutions, government bodies, industry stakeholders. Their aim is to typically guide researchers in distinguishing what is acceptable or not when conducting their research. In conducting this study, the researcher obtained ethics clearance from the University of South Africa Ethics Review Committee. This research enquiry was thus conducted within the framework of UNISA's ethics policy. Research subjects, in this case the engineers were made aware of their participation to a scholar study and anonymity of their feedback was ensured. Data collected was treated as private and confidential. Some of the data collected was extracted via screen grabs, consequently all information which might have jeopardize anonymity (real usernames, IP addresses etc.) was either removed or altered accordingly.

1.9 Thesis outline

Chapter One is an introduction to the current research. It presents the background context for the study and discusses issues related to the software engineering world as well as the South African software industry. It then enunciates the research problem, questions and objectives underpinning the study.

Chapters Two and Three present a two-part literature review for this thesis. The first layer which is presented in chapter two, covers a discussion ranging from the origins of the software crisis, to traditional quality issues in software engineering. The second layer in chapter three, contains an exploration of agile approaches to handle software projects with supporting methodologies, as well as their enabling technologies and tools.

Chapter Four contains a discussion on the selected research design, data collection methods and introduces the case study. The third chapter then describes enquiry paradigms as well as Activity Theory foundations underpinning the research.

Chapter Five draws from theory and methodologies previously described to introduce the conceptual framework driving this study. In this chapter, a software development industry case study is presented to understand how a local start-up tailors its software development agile processes and uses technology to ensure continuous delivery of quality software. The chapter then describes an overview of the design and implementation of the thesis' agile information system.

Chapter Six is the main part of the dissertation. In this chapter, data collected from semi-structured interviews, direct observations as well as experiences and views of the agile information system's users are discussed.

Chapter Seven proceeds with an interpretation of findings presented in the preceding chapter. It provides an explorative discussion of these findings and attempts to draw insights from quality delivery during the case study. The prototype implemented is further explored and its usage at the firm examined. The chapter concludes with a presentation of emerging theoretical prospects**.**

Chapter Eight concludes with an overview of the thesis at hand and proceeds to revisit research questions. In doing so, it reiterates some essential requirements to the implementation of a successful conceptual agile framework. Further down the line, it identifies areas of interest to explore without forgetting to note possible improvements on the proposed IS developed during the thesis case study. Final sections include a list of references, and appendices containing additional information on key concepts discussed during the current research exercise.

1.10 Summary

The first chapter of this thesis was a contextual introduction to South Africa's ICT sector and its local software industry. It introduced a background to the research problem and a history of the Software Crisis.

Chapter One also pointed out how despite a seemingly favourable macro-economic context, the issue of quality software projects delivery is still relevant to both local South African markets as well as academia in general.

A research problem was then enunciated. From this, the main research question and related sub-questions were established. To deal with resulting research objectives, the current thesis adopted a mixed methodology by leveraging literature surveys and a practical industry case study. During the case study in question, the researcher produced an experimental IS prototype for usage in software projects. The mixed methodology also included data collection techniques ranging from semi-structured interviews, to direct observations of participants as well as an electronic survey. Details of the research design adopted for this thesis are discussed further in chapter four. Consequently, before specifics of research methodology are further expanded upon, the next two chapters proceed to a literature review for the current research inquiry.

2. CHAPTER TWO: QUALITY IN SOFTWARE ENGINEERING

The current study suggests a framework for the delivery of quality software projects and products in tailored agile environments. To this end, Chapter two initially deals with the first sub-question of the research enquiry at hand: *What does quality entail in software engineering contexts and how does the modern software engineering industry deal with this issue?*

The chapter begins by presenting a detailed discussion on software engineering and related themes. Issues around software quality, some of its dimensions as well as its actual business value to organisations are subsequently addressed. Chapter two then presents a discussion on methodologies available to industry players and the advent of agile approaches in software engineering. After this, the literature survey then describes modern disruptive technologies available for organizations aiming to deliver quality technology solutions to market. These tools are presented to:

- Discuss the problems they solve.
- Show opportunities they provide to organisations in general and local start-ups more specifically.
- Identify shortcomings in their usage.

Figure 2.1 illustrates how the literature for this study was reviewed:

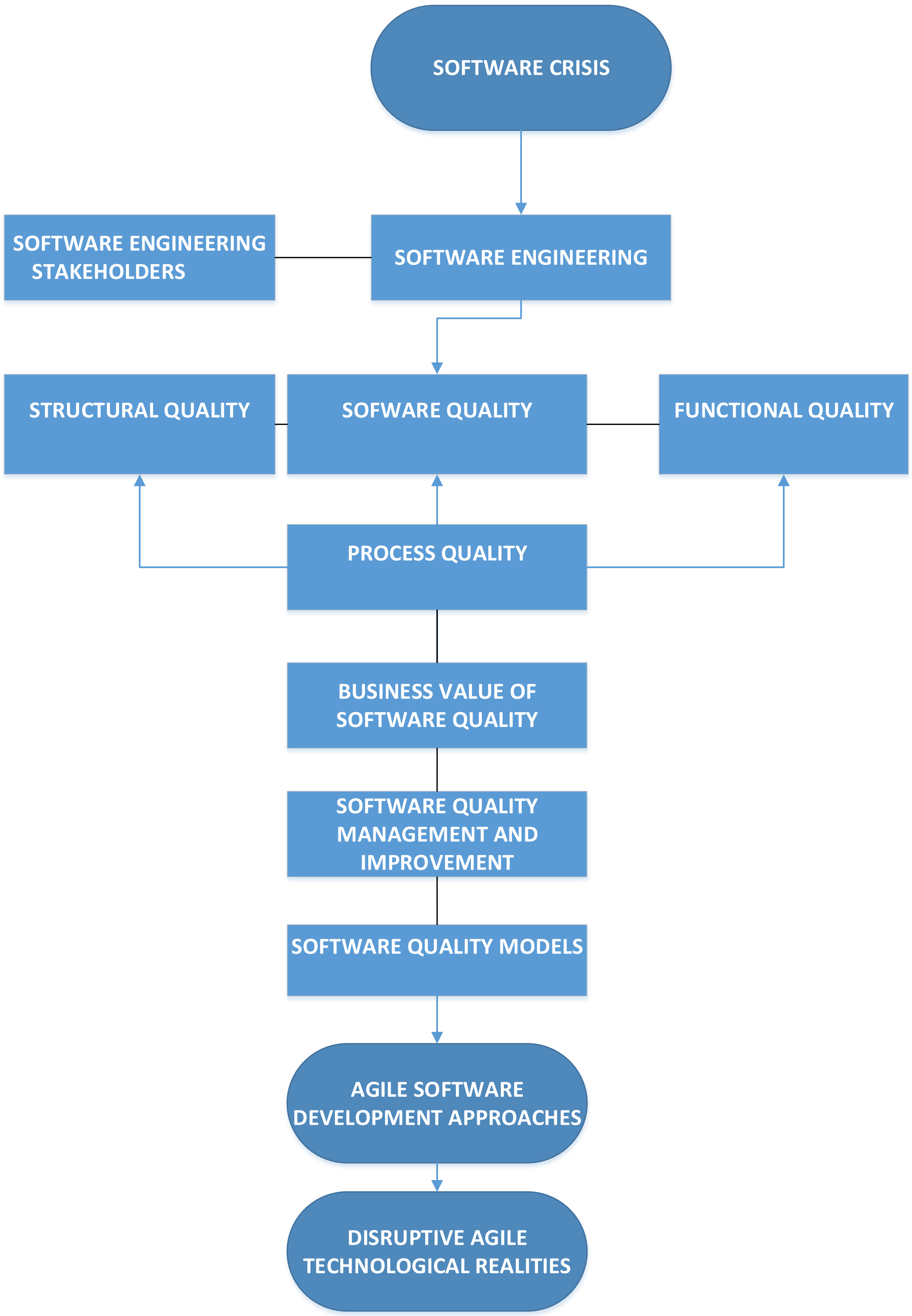

SOFTWARE CRISIS
SOFTWARE ENGINEERING STAKEHOLDERS
SOFTWARE ENGINEERING
STRUCTURAL QUALITY
SOFWARE QUALITY
FUNCTIONAL QUALITY
PROCESS QUALITY
BUSINESS VALUE OF SOFTWARE QUALITY
SOFTWARE QUALITY MANAGEMENT AND IMPROVEMENT
SOFTWARE QUALITY MODELS
AGILE SOFTWARE DEVELOPMENT APPROACHES
DISRUPTIVE AGILE TECHNOLOGICAL REALITIES

Figure 2.1: Literature review overview

2.1 The Software crisis

Following advances in computer hardware, it became apparent to the scientific community that new machines being built would require fundamental logic changes in order to achieve the expected improvements in terms of automation (Mnkandla 2009, pp. 1–3). However, due to the demand for new software reaching unmanageable levels, industry stakeholders quickly realised that a more formal and professional approach to software development was required (Mnkandla 2009, p. 3). This situation was fuelled by additional circumstances.

As software developments problems during projects emerged, issues ranging not only from the actual programming of code but also the difficulties encountered in the course of maintaining it appeared; this resulted in what became known as the *Software Crisis* (Mnkandla 2008, p. 1). While it was initially defined in regards to productivity, the software crisis was later identified as a major cause of problems in issues regarding software quality (Glass 1998 in Mnkandla 2008, p. 15). Academia has historically concerted and approached the origins of this software crisis from different angles (Brennecke *et al*., 1996, p. 3) :

- ***Technology imbalance theory*****:** During the 1960s, the fact that improvements in processor speed and memory had little impact on software techniques was a cause of the software crisis.
- ***Expectation theory:*** In the 1960s still, industry wondered if the software crisis was perhaps not an expression of the difficulty in meeting heightened expectations.
- ***Professionalization:*** The quest for software writers to professionalize themselves was itself investigated to address the software crisis.
- ***Economic:*** The economic driving force on the part of suppliers and customers who wanted to have control of pricing and delivery timetables was also considered as contributing to the software crisis.
- ***Dramatic Failure Theory:*** Academia explored whether the so-called software crisis was simply the result of few large-scale failures or catastrophes or rather a general accurate depiction of an unreliable industry.
- ***Dissemination issue:*** Academia also explored the extent to which government bodies and special interest groups could have influenced the identification/creation of an artificial software crisis.

- ***Labour perspective:*** The need for labour markets to commoditize software engineering skills as opposed to relying on scarce talented engineers was also considered a huge contributing factor to the software crisis.

It is with these problems in mind that new a discipline described as "*Software Engineering*" was born. Ascribed to a 1968 conference held under the hospices of NATO in Garmisch, Germany, software engineering revolves around designing, implementing, maintaining and documenting information processing systems (Mnkandla 2012, p. 279). A key objective of this new discipline would be to propose new development methodologies to address software project failures in terms of costs, timely delivery as well as defects management to improve quality delivery (Masrek, Hussin & Tarmuchi 2008, p. 137). The next section discusses software engineering in more detail.

2.2 Software Engineering

As implied from the paragraph above, engineering in the software world typically entails designing, implementing, maintaining and documenting information processing systems. Despite there being a general common understanding of the term among professionals, various definitions are found in various literature sources. It is often described as a discipline which applies scientific methods in the execution of software development (ISO/IEC & IEEE 2010, p. 331). Others define it as a discipline tasked with providing customers high quality software solutions while remaining within acceptable time and costs margins (Lethbridge & Laganiere 2004, p.6). One of the essential points which seems to emerge from literature discussing software engineering as a discipline, is that it is a key contributor in reducing failure rates of software projects. For instance, one of the main selling points found in academia in justifying the importance of software engineering is that it facilitates the ability to produce software on time and that it usually does this at a defined cost with high levels of quality (Humphrey et al. in Steward 2007, p. 6).

The definitions described above encompass the essence of software engineering. The case study component of this thesis is based at a South African local start-up, GZ Consulting Services (pseudo name), this study hence adopts the company's definition which accepts software engineering as an *"adaptive and agile application of engineering processes for the delivery of innovative software solution to customers"*. In the next section, an overview of software engineering goals is presented.

2.2.1 Software Engineering goals

While software engineering aims to encourage the development of products which are reliable, maintainable, efficient and reusable; it also helps companies by bringing organisation and methodology to the process of developing software (Dwolatzky & Trengove 2002, p. 7). Software engineering has traditionally put an emphasis on sound design practices, rigorous testing as well as comprehensive documentation (Shelly & Rosenblatt 2012, p. 508). This is done to facilitate the implementation of sound engineering practices which will contribute to deliver reliable products to end-users (Bauer in Giblin 2012, p. 4).

Ultimately, the goal of this discipline is to produce quality systems which will fulfil the purposes for which they are built. To achieve this objective, many contributors come at play. The next section hence discusses the participants in traditional software engineering projects.

2.2.2 Software Engineering project stakeholders

Software engineering projects can range from simple one-man jobs, to complex multi-layered endeavours. In this light, it is important to understand who the stakeholders that are to be involved in such projects are and their roles.

Software engineering project stakeholders have different roles and motivations. As such, they can be classified according to four major categories (Lethbridge & Laganiere 2004, p.10) :

- *Users:* This term refers to individuals interacting with software products as a key function of the daily business tasks required from them. They typically want a system that is easy, simple to use and enhances their productivity.
- *Customers*: These stakeholders are sometimes also known as the clients. They make business decisions such as authorizing the type of software that will be purchased from vendors and the intended users. Their main aim is to run the business more effectively and to increase its profits. Customers typically want software that helps their organisation make or save money.
- *Software Developers:* These stakeholders are also known as software engineers. They are the people who will write the source code for the actual software.
- *Development managers*: This term is used to describe management leadership in software firms and organisations. They often come from an educational background in business administration and have the responsibility for customer satisfaction.

In the same vein of categorisation, one can also identify three global types of stakeholders. These would include the *end-users* utilizing the software system, the *engineering team* which programs and implements the software as well as *project sponsors* responsible for financing software projects or products purchase (Chappell 2012b, p. 3).

Software engineering project stakeholders are driven by the fact that their endeavours ultimately must deliver a functioning system. The project participants understand that poor quality is the downfall of any enterprise, thus, emphasis must be placed on delivering a quality product to the target audience. It is vital for companies that need to remain competitive in today's knowledge-based economy that software products they deliver be as error free as possible. In this light, the next section proceeds to introduce software quality, the aspects or perspectives through which it can be understood as well as its business value to organisations.

2.3 Software Quality

Scholars have often found difficulties and many obstacles in establishing a generic definition of quality. This is mainly because it is a multidimensional concept that can encompass an entity, its viewpoint as well as the quality attributes of the observed entity itself (Kan 2002, p. 1). The bulk of ISO and IEEE literature describe the concept of quality as the sum of features which influence most a product's effectiveness in fulfilling the objectives it was built for. It can also be thought of as the complete set of features of a service or an artefact which influence its capacity to fulfil the needs for which it was developed (Samadhiya, Wang & Chen 2010, p. 320). A more concise definition of software quality would simply describe it as software that conforms to a customer's requirements (Kan et al. in Oster 2004, p. 39). The software engineering book of knowledge describes quality software as the satisfactory usage of software products in certain conditions by its intended users (Wallace & Reeker, 2001, p. 166). In actual software engineering projects, quality in software development is looked at as the combination of low defects with high user satisfaction levels (Jones 2010, p. 7). Research in the field mostly converges in correlating quality to the level by which customer expectations are met, the manner in which the system performs and the amount of defects or bugs it exposes (Sowunmi & Misra 2015, p. 867).

Jones (2010) also cites five important facts that illustrate software quality influence on industry:

- Software is viewed as the number one man-made culprit when major business problems occur.

- It is reported that poor software quality costs more than 500 billion dollars globally.
- Projects cancelled due to poor quality usually end up costing at least 15% more than similar successful projects.
- Software executives, managers and technical resources are sometimes considered as painful necessities as opposed to invaluable and respected assets by business executives. Even the position of CIO is considered (humorously perhaps) in some corners of industry as an endangered species.
- “Improving software quality is a key topic for the global ICT market”.

This eventually led to a troubling observation in a large of number of software engineering projects. In effect, many software development organisations do not pay enough attention into having a clear understanding of what software quality should entail. There is hence a perceived tolerance for defects in products that most engineering disciplines would not allow (Bessin 2004, p. 2). This can appear as problematic if one considers the risks organisations must mitigate when they do not pay attention to issues surrounding quality. It has been observed that there is a considerable incidence on costs and rework required when clients identify defects in software products which vendors have not eliminated before releases (Villasana & Castello 2014, sec. 3). This impact is amplified with the current surges in network traffic due to democratisation of mobile devices in contemporary societies (ibid., 2014). Also, there is a danger in tolerating poor software quality as it can in dramatic cases turn deadly. For critical software in the healthcare industry for instance, this can lead to serious injuries or loss of life (Samadhiya, Wang & Chen 2010, p. 320).

As such, organisations have a vested interest in steering away from disaster by understanding how to ensure the delivery of quality products. The United States (US) military for example contributed immensely in the quest to produce quality software. A study conducted for the US Air Force identified eleven common characteristics of quality software (Mccall, Richards & Walters 1977, p. 35). Also referred to as the *General Electric Model of 1977*, this model which gave birth to or inspired many of the more modern ones, ambitioned to bridge the gap between end-users views, expectations versus engineer’s perceived priorities during development and testing stages (Samadhiya, Wang & Chen 2010, p. 320). The model focuses on the capacity for software products to adapt to changes in their structure, their evolution regarding new environmental changes as well as the product’s operational characteristics (ibid., 2010).

A list of characteristics derived from this military research emerged from a specific subjective context; despite this, the characteristics identified would meet a consensus in most software engineering organisation. Table 2-1 illustrates these characteristics:

Table 2-1: The 11 characteristics of quality software (Adapted from McCall et al. 1977)

CORRECTNESS	*This expected quality feature describes the effectiveness of a software artefact in fulfilling the objectives it was built for.*
RELIABILITY	*This describes how reliable, correct and precise a software product's output is.*
EFFICIENCY	*This refers to the cost effectiveness in terms of lines of code (LOC) and computing resources a software product offers.*
INTEGRITY	*This refers to the level to which intrusions in a system can be monitored and controlled.*
USABILITY	*This refers to the system's user friendliness.*
MAINTAINABILITY	*A software product is deemed maintainable if the amount of time required to identify and patch code bugs while the system is operational does exceed the potential reward.*
TESTABILITY	*Testability ensures that not to great of an effort is required to evaluate the validity of tasks performed by a system.*
FLEXIBILITY	*This refers to the level of effort required for the adaptation of a system to evolving requirements*
PORTABILITY	*A system is deemed portable when the cost of porting the software to a different infrastructure is not prohibitive.*

REUSABILITY	*Software products which are usable by external applications and design with proper modularisation in mind are often the most reusable.*
INTEROPERABILITY	*This refers to amount of effort required to integrate one system to others external of internal pieces of software.*

The quality model summarized above illustrates how one needs to consider multiple facets when conducting a discussion on quality perspectives in terms of software engineering. These perspectives could exponentially vary depending on the organisation in question as beauty is essentially in the eyes of the beholder. There is hence no definitive or non-exhaustive list of what quality really entails in regards to the software engineering context (Lethbridge & Laganiere 2004, p. 19). Accordingly, modern studies have tried to arrive at more concise enquiries of the topic. A Microsoft sponsored study hence proceeded to examine what it described as the three aspects of software quality (Chappell 2012b, p. 3). Following figure 2.2, the next sections proceed to a presentation of these aspects; notably functional, structural as well as process quality.

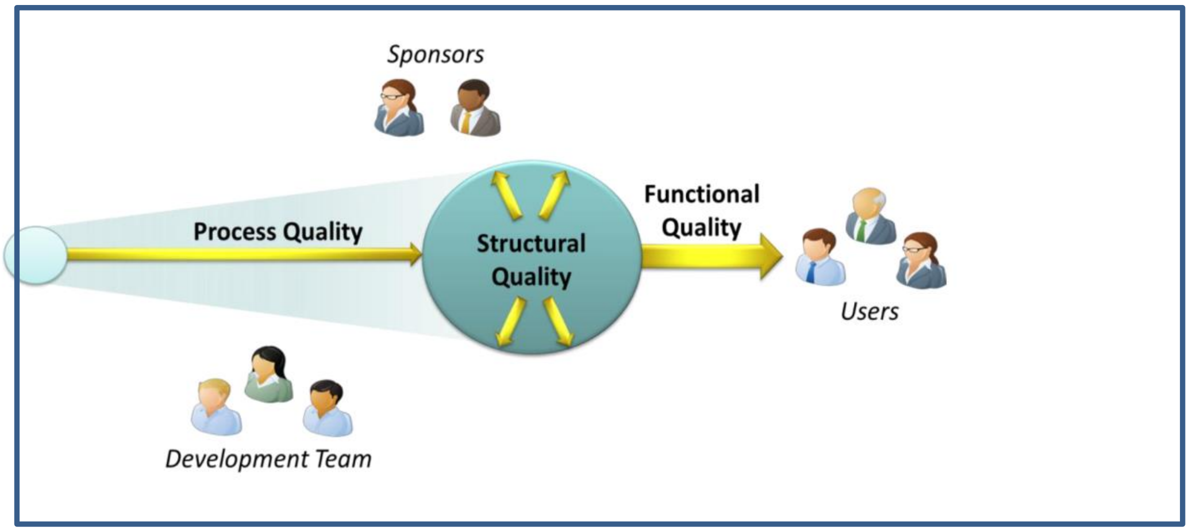


Figure 2.2: Aspects of software quality (Chappell 2012b, fig. 2)

2.3.1 Functional quality

Within the software engineering world, when mentioning functional quality in a business context, one typically refers to whether or the not the software products fulfils its intended functions in a satisfactory manner (Nagar & Thankachan 2011, p. 133). It ultimately entails understanding how well a particular piece of software or system complies after deployments when compared to the initial list of functional requirements it was intended to fulfil (Pressman in Liu *et al.* 2016, p. 167). In addition, one should consider the following attributes (Chappell 2012b, p. 3):

- *Conforming to requirements specified at project inception*: As requirements often change throughout a software engineering project, development teams need to understand and have a very clear idea about the requirements throughout to implement the software with accepted quality standards.
- *Creating software that has few defects:* Software defects need to be eradicated as efficiently and as early as possible in development process. The reason for this is that defects reduce expected reliability. Defects also have the adverse effect of rendering functionality vulnerable to security exploits and even at times totally hamper performance. While achieving zero defects is a holy grail that is unobtainable for most projects, software users tend to be even less lenient when they perceive the product they are using as unacceptably error prone.
- *Good enough performance:* Users of software will typically not accept an application which does not clearly explicated requirements.
- *Usability.* User experience needs to be optimal. An emphasis on User Experience is vital to retain customers' loyalty.

While software testing will commonly deal with the functional quality issues mentioned above, there is also a structural dimension which needs to be considered. The next section hence discusses the structure aspect of software quality.

2.3.2 Structural dimension of software quality

Structural dimension of quality in terms of the software world refers to the degree to which non-functional requirements are met (Nagar & Thankachan 2011, p. 133). It refers to the level at which the delivery of functional requirements is supported by the structural quality of a software product's source code (Liu *et al.* 2016, p. 167). When attempting to explain software structure, one usually describes the internal components of a system, their interfaces as well as the interactions between each and every one of them (Pan *et al.* 2010, p. 1202).

In a study on software reliability, the authors determined that a software product's inner structure is the foundation to the quality of its structural dimension; the objective is to evaluate if the product was developed with standard software architecture principles (Nagar &

Thankachan 2011, p. 133). Furthermore, while structural quality is harder to evaluate than functional quality, one could look at the following attributes when assessing structural aspect of software quality (Chappell 2012b, p. 4):

- *Testability*. Was the source code structured to facilitate testing?
- *Maintainability*. Was the code written in such a way that features can be added, removed or modified without introducing new bugs?
- *Code Understandability*. The level of complexity (appropriate or not) and readability of the code impact how quickly new software engineers to the project and effectively contribute to the source repository.
- *Efficiency*. Appropriate usage of resources in constrained environments.
- *Code security*. This feature is vital as software engineers need to structure their code in such a way that they cater for the risks associated with attacks such as SQL injection, buffer overruns, memory leakages etc.

While both structural quality and functional quality discussed previously are important, a third aspect is process quality. The next section discusses this.

2.3.3 Process quality

In engineering teams, a process typically refers to an ensemble of methods, activities and practices which are executed to implement and support software products; this whole also includes supporting functions such as project management as well as its related activities (Yinfen 2011, p. 2699). These processes have been shown to be critical factors of decisive outcome in terms of success or failure rates in software projects (Liu et al. in Öztürk, CİL & Zengin 2015, p. 2). Furthermore, to address the need for engineering projects to not overrun time and budget resources, organisations need to implement quality processes to ensure satisfactory overall quality for their customers (Koski & Mikkonen 2015, p. 1020). To grasp the importance of ensuring quality processes in engineering projects, one could perhaps initially introduce the concept of process modelling in the software world.

From an organisational point of view, business processes typically encompass a collection of documentable events, transactions and their outputs. To describe these processes, business and systems analyst can leverage techniques such as the Business Process Model (BPM) to graphically design an end to end business process flow (Shelly & Rosenblatt 2012, p. 10). Analysts can also approach this documentation of processes in more detail with more thorough standards such as the Business Process Modelling and Notation (BPMN) whose pillars are composed of symbols for activities, events, gateways and flows (Hoffer, George & Valacich 2016, p. 246). Models such as BPMN were designed to be intuitive and accessible to all enterprises users including engineers, business analysts as well as some end users (Desfray & Raymond 2014, p. 99).

In software engineering, process models are typically, representations of networked sequences that aim to eventually achieve software evolution (Scacchi 2001, p. 4). These sequences are represented by events, objects and transformations that inform the design of generic frameworks for project management in software endeavours (Lethbridge & Laganiere 2004, p. 428).

In practice, while these models are helpful to project managers in deciding what work should be done and the sequence in which this work should be executed; one should see these models as aids to thinking as opposed to set in stone instructions. Figure 2.3 below illustrates a typical bad application of a software process model called the opportunistic approach:

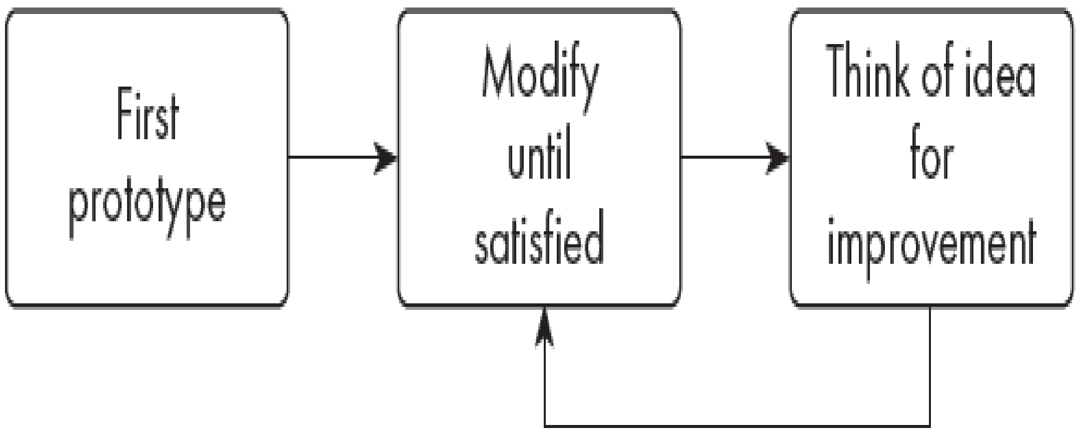


Figure 2.3 Opportunistic approach Model (Lethbridge & Laganiere 2004, fig. 11.1)

Organisations which follow this model usually encounter critical problems. With this approach, software engineers refine product prototypes until end-user satisfaction is achieved (Lethbridge & Laganiere 2004, p. 428); this approach to software development presents several limitations, notably:

- The fact that the opportunistic approach does not take into account of how critical clear initial requirements and detailed designs are. This could have an incidence on process quality because while the product could initially fulfil a limited set of requirements, user acceptance is dependent on many modifications on the software itself.

- The ad-hoc nature of this model will result in rapid quality deterioration if the initial design is as imprecise as it is bad.
- With this model, the need for rigorous testing is poorly understood. When a system eventually collapses due to this and all the facts mentioned before, there is simply no other choice but to start from scratch.
- The issues mentioned above ultimately drastically increase costs and sometimes run software projects into the ground.

The section above briefly described some software process models and the example of the opportunistic approach highlighted how choosing a wrong model could potentially have disastrous effects on a software project. From this point, quality of software processes which are the basis of software process models is discussed next.

In practice, applying appropriate software engineering processes influences the perception of end products, stakeholders hence have a vested interest in improving this aspect of quality. Industry literature hence described three principal attributes of software process quality (Chappell 2012b, p. 4) :

- *Meeting delivery dates.* In effect, process quality ensures that software features, artefacts are delivered in time by enforcing processes that are adhered to.
- *Meeting budget constraints*. Process quality also ensures costs do not go overboard and that financial constraints are clearly understood by the team and respected as much as possible.
- *An iterative development process which reliably delivers quality software*. True process quality is only achieved when it consistent an applicable from one project to the other.

While organisations strive to remain competitive, it is only when firms grasp how critical quality is to their survival that they understand it actual value. In industry practical use cases, organisations need to factor in quality during development of projects rather than await incidents to implement fixes in post-implementation. Some studies have for example suggested that in smaller sized projects, while teams should be concerned with implementation activities , leadership needs to also deal with quality planning throughout the entirety of a project's lifecycle (Lin, Zhenyu & Lizhi 2012, p. 4).

To reiterate the impact of quality for engineering project teams, the next section attempts to address why quality is so important to organisations from a business point of view.

2.4 The Business value of software quality

Organisations which value quality thrive due to several factors. This is because they become more innovative and responsive to an ever changing market place, increase their competitive advantage as well as ultimately succeed in reducing their total costs of ownership (Bessin 2004, p. 3). For example, an IBM sponsored study explained that for software development organisations or units to survive and succeed in the challenging business environment they operate in, these organisations need to place a heavy focus on quality (ibid., 2004). Concluding its analysis on the value of software quality, the study advanced that including quality-oriented processes in the software development lifecycle will ultimately promote innovation for organisations, thus lowering costs by increasing standards and predictability, reduce risks, eliminate rework and at the same time ensure that differentiation with other competitors is apparent to customers.

From the study cited above, one would retain that the business value of software quality eventually becomes more visible to organisations which understand that addressing quality issues will in the long term always cost less than ignoring them.

To further examine the business value of software quality, some industry literature suggests that one could isolate this issue in two separate parts, notably the business value of quality in *externally as well as internally* facing software (Chappell 2012a, p. 2). The next section discusses these two parts.

2.4.1 Business value of externally facing software

The term *externally* facing software refers to retail software products (Chappell 2012a, p. 2). A few examples can be used to illustrate the negative impact bad quality software can have on a business. For instance, in April 2011, a hacker group attacked Sony's PlayStation network. This intrusion resulted in the compromising of around 77 million accounts (Williams 2011, para. 2). What happened is that the attackers identified a known flaw in the database system security features and exploited it (ibid., 2011). Sony later estimated that the incident would cost the company about $170 million in the 2011 financial year. This is just one example to illustrate how organisations especially large companies risk damaging their business if they do not put an emphasis on software quality. Figure 2.4 shows the typical impact of low quality externally facing software on businesses:

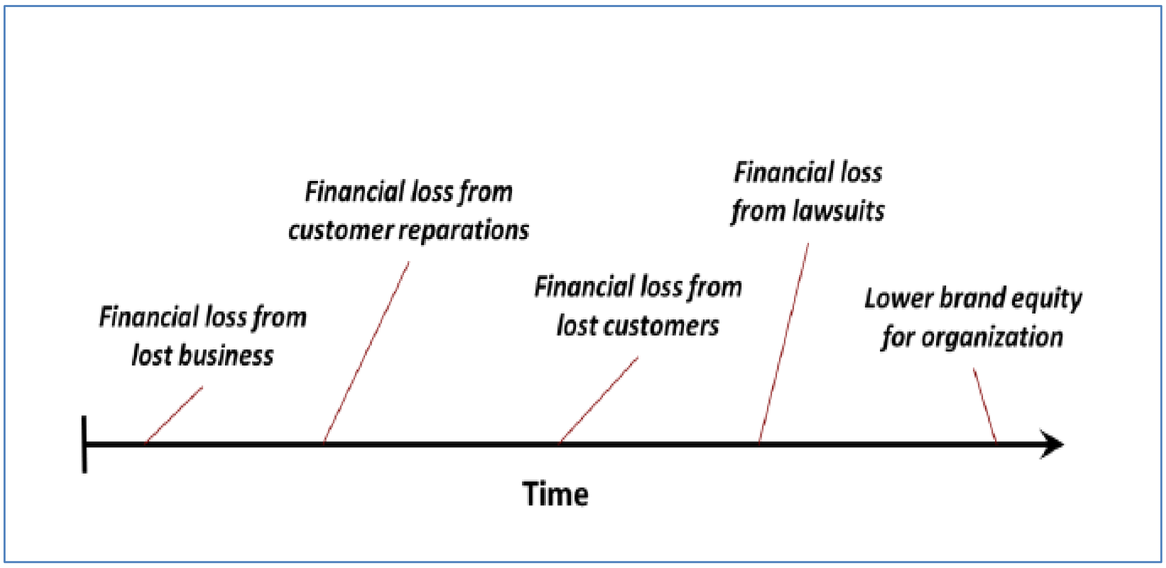

Financial loss from customer reparations
Financial loss from lawsuits
Financial loss from lost customers
Lower brand equity for organization
Financial loss from lost business
Time

Figure 2.4: Impact of low quality externally facing software (Chappell 2012a, fig. 1)

Figure 2.4 illustrates the need for an emphasis to be placed on delivering high quality to customers. Businesses which do not realise the importance of such issues for customers risk tremendous negative repercussions on their organisations, ranging from financial losses to actual lower brand equity over time.

2.4.2 Business value of internally facing software

The term *internally* facing software refers to software applications created by an organisation and primarily used by its own employees (Chappell 2012a, p. 4). The impact of low quality with such software also has a negative influence on businesses over time. Figure 2.5 illustrates this:

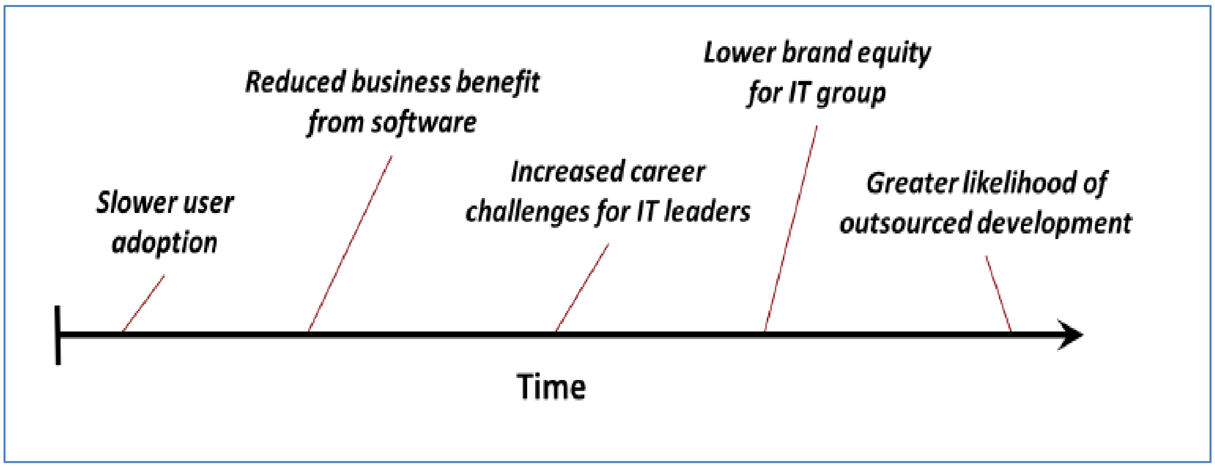

Lower brand equity for IT group
Reduced business benefit from software
Increased career challenges for IT leaders
Slower user adoption
Greater likelihood of outsourced development
Time

Figure 2.5: Impact of low quality internally facing software (Chappell 2012a, fig. 2)

Internally facing software when characterized by lower quality presents a host of potential hazards to organisations (Chappell 2012a, pp. 4–5):

- *Slower user adoption*: This usually happens when for instance a complicated business improvement process application is implemented. The obvious negative impact is that the objective of improving the intended process will take longer than if a high-quality application was implemented.
- *Limited business benefit*: This can occur even if user adoption was deemed smooth. The quality issues start popping up after the software is in use and bugs become more apparent to the users. This might for instance result in inaccurate reporting which could ultimately have a dramatic incident on any business.
- *Challenges on IT leaders' careers*: IT managers responsible for projects which delivered low quality software might end up tarnishing their reputation, and this could negatively affect their career paths in industry.
- *Lower brand equity for the IT group*: The IT Unit itself might end up being negatively perceived by the rest of the organisation.
- *The threat of outsourcing:* When a brand becomes so damaged that it risks reaching the point of no return, customers tend to flee. Technology firms are not exempt from this. Software Engineering organisations can only maintain their brand by consistently delivering quality software.

It hence appears that maximising the business value of software quality requires organisations to understand risks involved. In practice, the effects of the Sony 2011 incident could have perhaps been mitigated had senior leadership better anticipated the potential dangers of underestimating the need for thorough and continuous quality control. The organisation could have avoided the brand damage it suffered had it understood the correlation between an application's quality and the value it yielded for the business in terms of cost, time gains or simply user satisfaction.

While considering the overall business value of quality to most sectors, the ICT industry has understood that major problems in projects occur when companies try to directly apply quality lessons learned in manufacturing industries to the software development process (Chappell 2012b, p. 2). To further illustrates this, the next section proceeds to a discussion of particularities of the software development world.

2.5 Specificities of the software engineering world

Transferring quality lessons learned during manufacturing processes to the software development world often yields more problems than solutions (Basili & Caldiera 1995, p. 55). This tends to happen when software engineering executives fail to note that manufacturers will typically develop their quality models by retrieving large amount of data from business units where processes are repeated in an almost mechanical way (ibid.,1995). The incompatibilities in philosophies are most evident if one understands that software development has traditionally often heavily relied on specific intellectual capital which needs to be adapted to the problem at hand. It is hence sometimes difficult to replicate manufacturing models' dependence on massive repetitions of the same processes to a practical software development use case (ibid.,1995). In practice, while for example, a civil engineer will pick up most design flaws on a construction project very early in the design phase, this is not always possible in software projects. A known issue with software engineering for most professionals in the information technology field is that many times, some design defects are only picked up by engineers once the software has been implemented. To further illustrate the idea mentioned previously one could consider the following version of Murphy's Law which states (Maskelyne & Devant 2013, p. 54):

"*It is an experience common to all men to find that, on any special occasion, such as the production of a magical effect for the first time in public, everything that can go wrong will go wrong. Whether we must attribute this to the malignity of matter or to the total depravity of inanimate things, whether the exciting cause is hurry, worry, or what not, the fact remains"*.

One could summarize it as follows: "*If anything can go wrong, it will".*

To contextualize the quotes mentioned above, one could use the example of an experienced software engineer. In, practice, many such professionals could testify that often, certain bugs tend to pop up only after systems go-live. Older studies had for instance suggested that about only 50% of defects can be successfully identified by testing (DeMarco in Head 1994, p. 48).

This used to happen because production environments have traditionally been very difficult to replicate perfectly during development and de-bugging, for these reasons, unforeseen program behaviours caused by unplanned utilization by end-users are not uncommon in the software engineering world. A mathematical approach to further understand this problem provides another theoretical lens.

In effect, the behaviour of software which contains a certain amount of bugs is described in Poisson survival statistics (Littlewood in Brady Anderson & Ball, 1999, p. 5).

To illustrate this, University of Cambridge research explains that the probability that a software bug stays unnoticed after running tests*(t)* can be expressed in the equation: $pi = e^{-Eit}$ Ei represents the portion(also referred to as the *virility)* of software bugs dependent on the input space it influences (Brady, Anderson & Ball 1999, p. 5). The inference this gives us is that even when planned or random testing is executed in software engineering projects, there will always remain a certain number of undetected defects.

While this equation is applicable to simple systems with relatively few bugs, extensive empirical studies have shown that if one applies this mathematical model to large and complex systems, tweaks are necessary to the equation as the odds of failure for the t-th test isn't correlated to e^{-Et}. It is related rather related to k / t for constant k (Brady, Anderson & Ball 1999, p. 5). The Cambridge study acknowledges these limitations because in instances where complex systems have been observed to have mean times to failure of 20000 hours, equivalent duration testing is required for the equation to fit the paper's narrative (Butler & Finelli in Brady, Anderson & Ball 1999, p. 5).

When attempting to draw parallels between the software engineering world and biological species, researchers apparently comfort Murphy's Law by positing that the amount of software bugs which remain latent despite the implementation of selection processes is maximized (Brady et al. 1999, p.6).

Ultimately, what the study shows is that although biological species adapted during evolution to changing environments at minimal cost in early deaths, in the software engineering world, testing can only remove a certain number of defects which also need to be consistent with the tests applied anywhere; this is obviously a negative for software engineers as it means that even after testing, a large number of latent bugs in software can in fact remain problematic *(ibid.,* 1999). At this stage, to summarize the current discussion, two facts seem to emerge:

- Software engineering development processes are very different from typical manufacturing processes; quality models derived from successes in one area are hence not necessarily applicable to the other.

- When considering Murphy’s Law in a software engineering context, one understands that not only do some defects only appear after systems are implemented, but additionally; mathematical models tend to show that even after rigorous testing during the software development process, certain defects tend to remain latent until unforeseen circumstances occur.

Taking into account the unpredictable nature of what could happen after a software project is deployed despite careful planning, testing and execution; software engineering stakeholders need to hence put emphasis on quality and quality related issues throughout their entire software development lifecycle, to this point, quality management is discussed in the next section.

2.6 Quality Management and Software Quality Improvement

2.6.1 Background to quality management

Industry variables such as the global market place and contemporary consumerism have largely contributed to a shift in the perception of quality, its role and impact on a firm’s competitive edge (Cohen 2008, p. 28). Efficient modern organisations understand the need to remain competitive in order to survive. To achieve this, firms need to constantly improve quality in every aspect of their internal activities for them to produce quality goods and services on markets (Shelly & Rosenblatt 2012, p. 508).

This has led to an ever-increasing need for higher quality products from organisations and consumers alike; businesses are therefore forced to strive for a competitive advantage based on product or service quality (Cohen 2008 p.136). Quality management (QM) is hence made even more relevant to both industry and academia because of its impact on business performance. Key drivers which motivate modern organisations’ increased interest in putting an emphasis on QM include the desire to increase profits while maintaining a competitive edge on local and international markets in a globalised economy (Ghanim, 2016, p. 6). Against this backdrop, quality management can be explained as the coordination of activities aimed at directing and controlling an organisation in regards to quality (ISO/IEC & IEEE 2010, p. 288).

As an introduction to grasp quality management basics, it is often recommended that organisations focus the core of their efforts on measuring and evaluating business processes to accurately identify gaps between customer requirements and actual product functionality (Deming 1993, chap. 2).

Initially inspired by work previously conducted for quality research at Dell Labs, Deming (1993), further proposes that organizational processes consistently leverage feedback loops which would enable management to identify and accordingly modify processes which need improvement. To this light, the author describes a quality management cycle he calls *Plan-Do-Study-Act"* or PDSA (Deming 1993, chap. 6). The PDSA features the following steps:

- *Plan:* This suggests that organisations should start using knowledge gained in previous experiments for them to plan studies which may be expected to have help process improvement.
- *Do:* This refers to execution of studies designed during the planning phase.
- *Study:* This refers to the fact that organisations must carefully evaluate and understand the results obtained after previous steps.
- *Act:* When the studies yield positive results, organisations should proceed to implementation and use the studies as a basis for continuous research and development with the PDSA lifecycle. In the event where expected results are not obtained, management needs to go back to the drawing board and re-enter the planning phase of the PDSA lifecycle with the objective of developing a novel approach.

Deming (1993) illustrates this PDSA cycle:

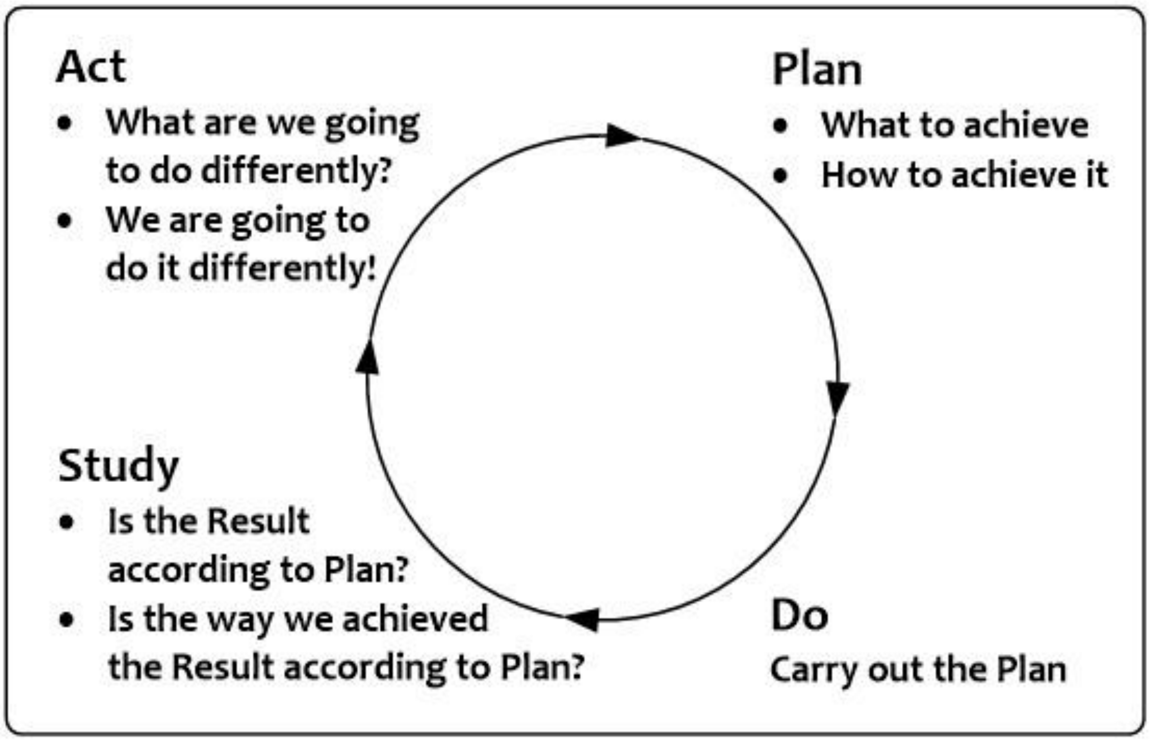

Act
• What are we going to do differently?
• We are going to do it differently!
Plan
• What to achieve
• How to achieve it
Study
• Is the Result according to Plan?
• Is the way we achieved the Result according to Plan?
Do
Carry out the Plan

Figure 2.6: Plan-Study-Do-Act (PDSA) cycle adapted from Deming (1993)

The essence of Deming's model suggests that to truly grasp and apply quality management, organisations need not only focus on understanding their core business processes, but they also need to continuously engage in re-planning, study, and tweak these processes.

2.6.2 Quality management in software engineering

In many industry settings, quality management mainly serves as an umbrella activity which is ongoing throughout the software development lifecycle and which ensures that development is done appropriately as opposed to re-done countlessly because of quality issues (Pressman & Maxim 2015, chap. 21). Models derived from quality management concerns help organisations improve the monitoring and handling of software quality while projects are ongoing (Kan 2002, sec. 9.9). This enables the reduction of the amount of rework, which in turn results in a reduction of costs and improved project delivery timelines. To further summarize the key points of quality management in a software engineering context, Pressman and Maxim (2015) explains that essential features need to be effective, notably:

- ➢ An efficient quality assurance process.
- ➢ Specific quality control tasks such as formal technical reviews and for instance a multi-tiered testing strategy.
- ➢ Effective version control and verifications systems for source code.
- ➢ The implementation of software engineering standards.
- ➢ The delivery of quantifiable measures and reports.
- ➢ Effective engineering practices, tools and methods.

Considering the points of quality management listed above, this thesis considers that a symbiosis of these key features from an engineer's perspective form an essential pillar to the quest of gaining a competitive edge at the very least in terms of product quality.

In software engineering project settings, contributing to the organisation's competitive advantage equates not only to producing new and innovative products, but also to ensure that each new release does not reach customers before acceptable quality standards are met. In this light, the next section discusses quality assurance as well as software process improvement.

2.6.3 SQA and SPI

In many organisations, IT departments will often leverage the services of a quality assurance team (QA). This team is tasked with testing software changes to all applications and establish reviews as well as the reporting of its findings to ensure engineering projects rectify what needs to be fixed (Shelly & Rosenblatt 2012, p. 29). This Quality Assurance is important regardless of the methodology being used. When used in traditional waterfall models, quality assurance is usually done after products have been developed; in agile teams, because of the dynamic nature of the software artefacts engineers need to produce at regular intervals, QA should ideally to be adapted to bring stability and quality during each iterative phase (Bhasin 2012, p. 66).

In the context of engineering projects for modern organisations, Software Quality Assurance (SQA) encompasses an established evaluation mechanism leveraged by teams to monitor quality control processes while enforcing compliance to required formal standards in a continuous manner (Sowunmi & Misra 2015, p. 868). While SQA considers that processes in the value chain form part of a bigger package that needs to be delivered to consumers, Software Process Improvement (SPI) considers from its perspective that engineering processes are actually the central deciding factor which organisations should focus on when attempting to deliver quality software artefacts (Ashrafi 2003, p. 670).

In engineering projects, process improvement often refers to a term coined by the Software Engineering Institute (SEI) from Carnegie Mellon to describe a framework leveraged by CMMI models for the integration of software engineering activities in projects and their expected end-user artefacts (Shelly & Rosenblatt 2012, p. 509). Organisations which have opted to leverage models such as CMMI place a specific interest in quality management and assurance. These businesses have typically also adopted Software Quality Assurance and often have dedicated SQA groups or teams responsible for guiding the enterprise in all of its process improvement related endeavours(Kan 2002, sec. 17.10).

As such, to ameliorate the management of software engineering activities, organisations can leverage Software Process Improvement (SPI) to better analyse their internal development processes and proceed to improve them (Kuhrmann *et al.* 2016, p. 89).

2.6.4 Software quality improvement

The value chain theory is an interesting lens one could use to discuss software quality improvement. For an organisation, the ability to perform particular activities and effectively manage the links between these activities equates to obtaining a competitive edge (Porter in Wiggins 1997, p. 1). Furthermore, chain theory examines an organisation’s processes through each step of the production process as value added to the final product. In this light, if one considers that improvements made at each step of a value chain could improve the entire chain, a clear link can be established with the current research's general theme. This as the current thesis's central theme assumes that improving the process by which software is produced will ultimately improve the software product's quality and hence perhaps yield a competitive advantage for the organisation.

At this point of the study, a relation between software quality improvement, quality management and software quality emerge. In effect, the current chapter presented Process Quality as a key aspect of overall product quality. This is supported for instance by research which highlights effective software engineering practices, tools and methods as an essential feature of effective quality management in a software engineering context (Pressman & Maxim 2015, chap. 21). The next section hence proceeds to a discussion on the history and importance of methodology in software engineering.

2.6.5 Software Engineering methodology as a foundation to quality

The Software Development Life Cycle consists in detailing a plan to manage software projects. It has traditionally been one of the most popular approaches to running software projects due to the formal structure it provides in software engineering teams aiming to deliver products within set budget and time constraints (Larman et al. in Patwardhan *et al.* 2016, p. 1).

The methodology SDLC suggests also aims to facilitate quality in development processes. Below is a typical representation of the various stages of the SDLC:

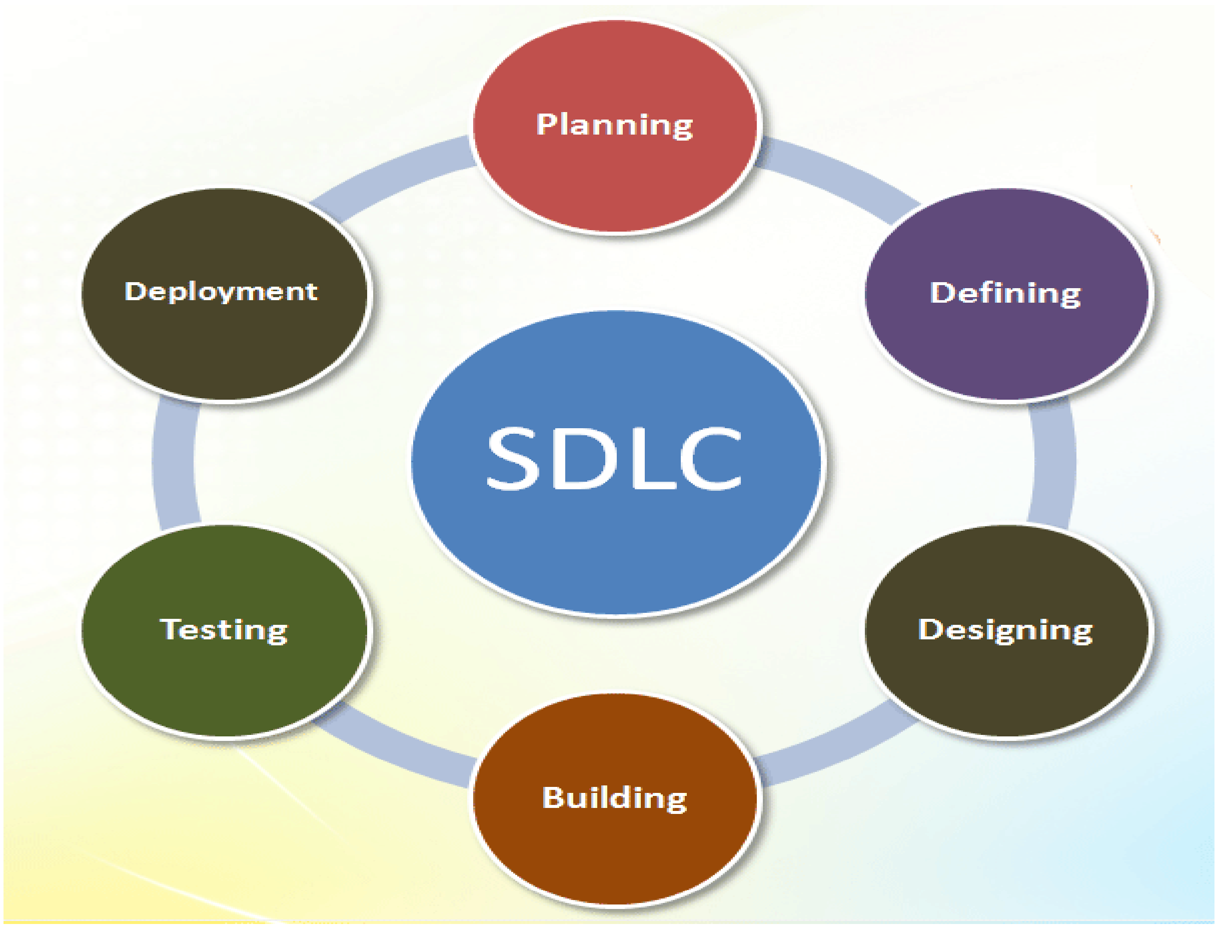


Figure 2.7: SDLC phases

Spawned from the SDLC approach, a methodology called the Waterfall lifecycle was introduced as a set of linear and sequential phases to execute software projects (Bell & Thayer 1976, p. 67). Aimed at providing a development framework which would provide some formal structure for software project teams, it comprises processes or steps which need to be executed sequentially (Giblin 2012, p. 3). The waterfall approach to software project management revolves around a simple methodological pillar: each phase of a project lifecycle is dependent on the preceding step, hence no phase can commence before the previous one is complete (LucidChart 2017, para. 3). It is usually understood to be the traditional foundational benchmark for running software projects in a top-down manner (Thummadi, Shiv & Lyytinen 2011, p. 68). Figure 2.8 illustrates this model :

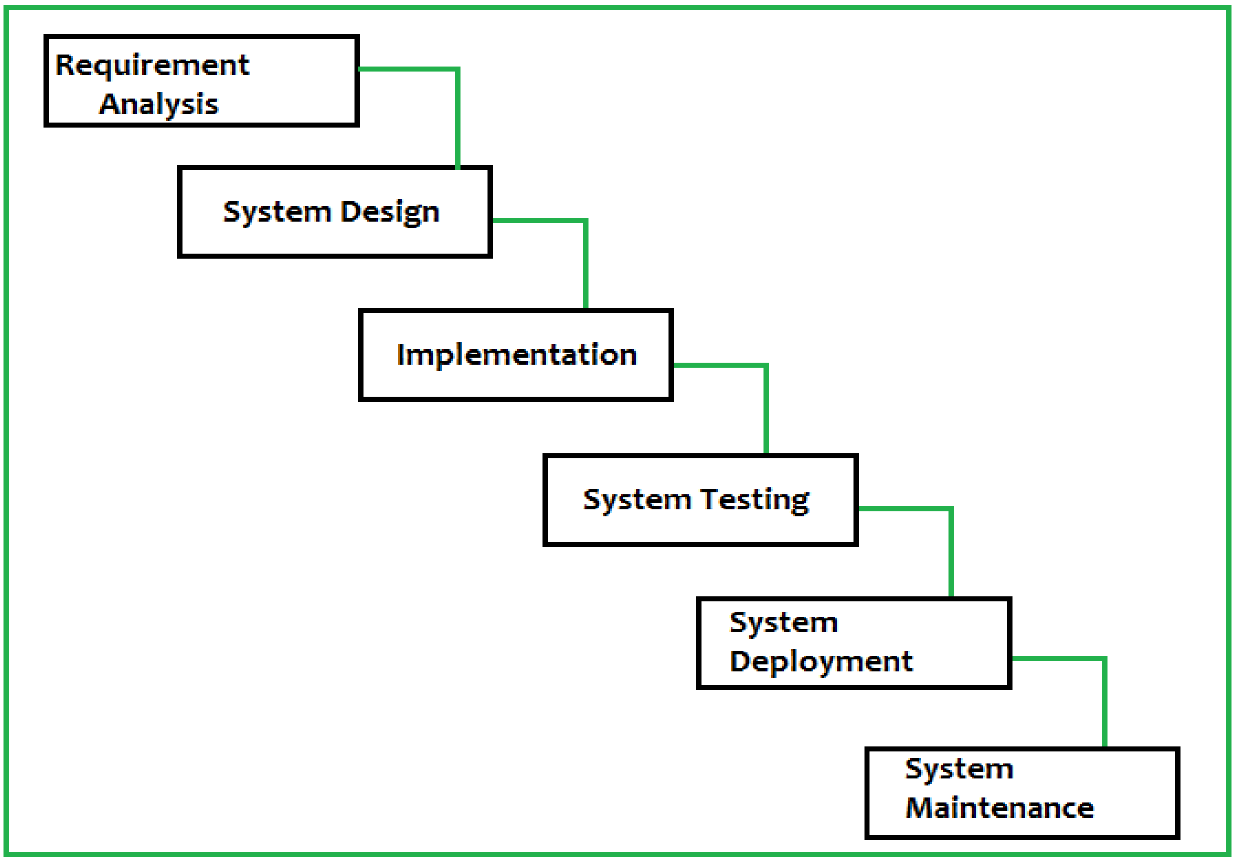


Figure 2.8: Waterfall Model adapted from(Boehm & Turner in Mnkandla 2008, p. 15)

A detailed discussion of the concepts introduced above are beyond the scope of this section. What is interesting to note though is the emphasis traditional SDLC places on software engineering processes. Moreover, despite the SDLC's strong points, the sequential thinking rooted in this paradigm constitutes a limitation as it is predictive in nature and assumes that business requirements can accurately be anticipated (Mnkandla 2008, p. 16).

This research ultimately ambitions to propose a practical agile oriented framework supported by related technological tools for the continuous delivery of quality software. Before presenting discussions on agile methodologies, the next sections hence proceed to introduce software quality models.

2.7 Software Quality Models

Though many quality models have emerged from sectors with no apparent direct link the ICT industry, some have successfully been created and implemented because of real world software engineering problems. This has led to extensive research in the field yielding improved quality models.

With regards to gaining insight into software quality, researchers have built models which rely on metrics to improve it over time. In order to proceed to a concise synthesis of software quality models available to organisations, six models deemed current or instrumental in having inspired more recent models are discussed (Acton, Kourie & Watson 2014, no. 52).

2.7.1 Hyatt and Rosenberg's quality model

In terms of quantitative approaches to quality in software projects, previous research has shown that rigorous estimations and evaluation metrics can yield immense benefits. One of these approaches is Hyatt and Rosenberg's quality model. By placing an emphasis on metrics which are meant to facilitate risk assessment for software projects in the space industry, quantitative techniques are argued to help improvement of quality in projects (Hyatt in Liu, Kane & Bambroo 2006, pt. 1.2). The importance of risk assessment is capital in the context of scalable distributed software architectures (Lima 2010, p. 349). At its introduction, Hyatt and Rosenberg's research revolutionized software engineering teams 'approach to risk assessment and management for the improvement of quality of their projects.

This software quality model aimed to apply traditional concepts to the needs of project managers at NASA and the Goddard Space Flight Centre (Hyatt & Rosenberg 1996, p. 209). Considering that the poor gathering and documentation of requirements had an impact on functioning software, researchers elaborated Hyatt and Rosenberg's model to identify and mitigate potential risk areas (Wilson, Rosenberg & Hyatt 1997, p. 161). This model is based on the perspective of a project manager who leverages certain metrics which are then used to provide an indication of the risk in each risk area (Acton, Kourie & Watson 2014, p. 2). The creator of the models establish four goals from their understanding of software quality (Hyatt & Rosenberg 1996, p. 209) :

Table 2-2: Hyatt and Rosenberg's quality goals

REQUIREMENTS QUALITY	*This goal aims to produce requirements evaluated by the following attributes:* • *Ambiguity* • *Completeness* • *Understandability* • *Requirement Volatility* • *Traceability*

PRODUCT QUALITY	*This goal aims to produce code and documentation that will be evaluated by the following attributes:* • *Structure/Architecture* • *Reuse* • *Maintainability* • *Documentation adequacy*
IMPLEMENTATION EFFECTIVITY	*The objective here is appropriate usage of resources within project time, resources and costs constraints.*
TESTING EFFECTIVITY	*This objective aims to facilitate the timely identification of faulty software components and remediate to these while ensuring the software will behave in an acceptable manner post go-live.*

2.7.2 Cleanroom software engineering

Introduced because an efficient, effective and rigorous disciplined approached to executing software projects successfully had not been yet established, Cleanroom software engineering was enthusiastically welcomed by organisations and academia alike as a possible approach to improve quality in software development projects (Beizer 1997, p. 14). It was introduced at IBM in the early 80s as a group focused model aimed at ensuring usage of sound software engineering processes and related practices (Deek, McHugh & Eljabiri 2005, p. 31).

While its ambition was to yield higher quality software products with a zero-tolerance policy for defects, it leveraged formal methods such as correctness verification and object-based box design while heavily relying on statistical controls to produce its promised benefits (Linger 1993, p. 2). Projects advocates for Cleanroom methodologies argued for a combination of formal specifications and the leveraging of evolutionary development techniques (Larman & Basili 2003, p. 53).

It can be described as a methodology which aims to enable high software reliability under statistical quality control. In this model, the reliability of the software is measured by its Mean Time to Failure (Acton, Kourie & Watson 2014, p. 2). Studies have suggested that software users are more affected by a software system's reliability than the number of defects it might have (Currit et al. in Acton, Kourie & Watson 2014, p. 2).

The assumption this theory-based engineering approach has is that output of the development phase of software production is of high quality. Hence, there should be more focus on testing the software's reliability. This is made effective by leveraging on one hand a mathematical foundation during the specification, design and quality check phases, and on the other hand by making use of function- theoretical methods to ensure software correctness as well as ensuring statistical usage testing to certify product quality (Acton, Kourie & Watson 2014, p. 2).

2.7.3 Source code management approaches

A source control management system provides file management and version control to ensure software engineers do not overwrite each other's changes. Models inspired by SCM based approaches propose an automated evaluation of metrics based on a software product's source in order to analyse its quality (Barkman et al. in Acton, Kourie & Watson 2014, p. 3). Organisations such as Google for instance leverage a consolidated source control system which is used by all engineers at the company except in open-source type of projects which are typically shared with a wider audience (Henderson 2017, p. 3).

The wide adoption of products such as Subversion, Microsoft's Visual Studio team, Git based systems etc. seems to attest to the pertinence of such approaches to the software engineering community.

2.7.4 ISO Standard

ISO is a general standard for all industries which lays much focus on an organisation's quality's system. The 2004 iteration of ISO standards (ISO 9000-3), proposed a framework for quality assurance in the production and maintenance of software artefacts (Shelly & Rosenblatt 2012, p. 510). This standard typically requires that software engineering organisations and project teams implement a detailed step-by-step plan which details the full lifecycle between gathering user requirements and the delivery of end user products (*ibid.*2012). Traditionally speaking, ISO standards suggest that when an organisation implements, maintains and continually improves the processes used in the production of its products and services, the output of these processes will be of consistent or improving quality (Acton, Kourie & Watson 2014, p. 3). Subsequently, one of the first thing companies need to focus on is to identify in early stages which of software engineering processes require improvement (McManus & Wood-Harper 2007, p. 315).

In terms of software quality specifically, ISO/IEC 25010 proposed a generic quality model which focused on usage quality by isolating different quality in use components. These components include a satisfaction in use aspect, and effectiveness as well as an efficiency in use aspects (Hussain & Mkpojiogu 2015, p. 10).

For smaller organisations which have traditionally struggled to implement previous standards which they didn’t consider applicable to their use cases, ISO/IEC 29110 is increasingly attractive to adopt as it was designed and geared specifically towards the needs of smaller sized engineering houses (Larrucea *et al.* 2016, p. 85).

2.7.5 Capability Maturity Model for software engineering

Similarly, to ISO 9000, the Capability Maturity Model advocates for commitment by management in driving a quality-oriented framework in running their business processes. It proposes basic foundational pillars which include the establishment of basic management controls, clear understanding of processes, tangible quality standards which require rigorous metrics to be evaluated and ultimately, a continuous process improvement culture (McManus & Wood-Harper 2007, p. 322). Its most recent emanation is described as the Capability Maturity Model Integration (CMMI). CMMI advocates for extensive formal pre-planning to produce quality outcomes by relying on predictable checklists and documentation to ensure consistency throughout an organisational value chain (Giblin 2012, p. 5).

The model also encompasses a training component for improving processes in organisations as well as an evaluation methodology and service provided by the Carnegie Mellon University. Having proven its effectiveness in many organisations worldwide, CMMI attempts to introduce process improvement in organisations by tracking and monitoring processes to better them through five distinct maturity levels (Shelly & Rosenblatt 2012, p. 509). Figure 2.9 illustrates these CMMI maturity levels:

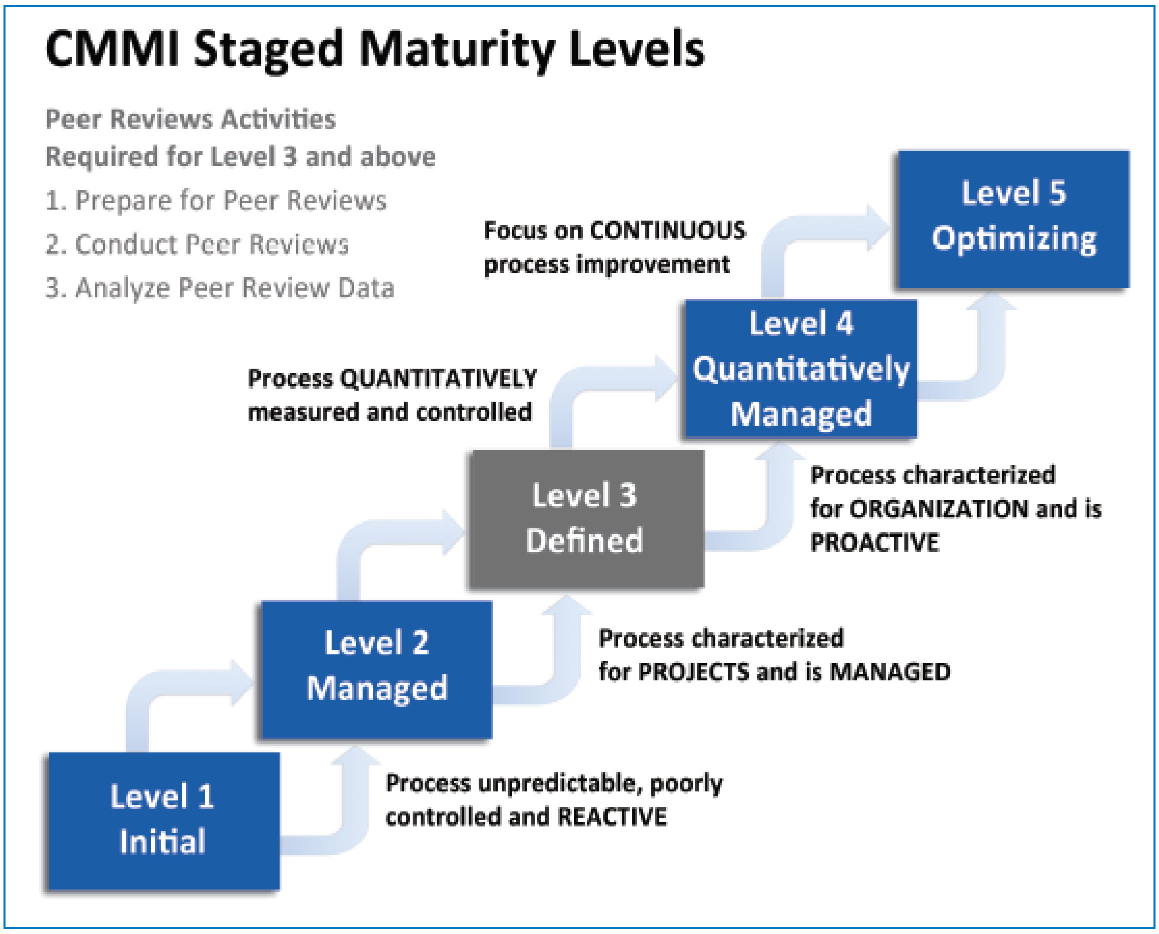

CMMI Staged Maturity Levels
Peer Reviews Activities
Required for Level 3 and above
1. Prepare for Peer Reviews
2. Conduct Peer Reviews
3. Analyze Peer Review Data
Level 5 Optimizing
Focus on CONTINUOUS process improvement
Level 4 Quantitatively Managed
Process QUANTITATIVELY measured and controlled
Level 3 Defined
Process characterized for ORGANIZATION and is PROACTIVE
Level 2 Managed
Process characterized for PROJECTS and is MANAGED
Level 1 Initial
Process unpredictable, poorly controlled and REACTIVE

Figure 2.9: CMMI Staged Maturity Levels

The CMMI approach to quality aims to influence organisations by establishing process areas with specific objectives and the required practices to achieve these goals. Another way to look at CMMI is to understand it as a set of products bundled together into a management suite. This suite is aimed at the facilitation of process improvement by leveraging integration and complementarity between various Capability Maturity Models otherwise labelled as CMMs assimilable to process areas: Software, Systems Engineering and Software Acquisitions (Omran 2008, p. 2).

For project teams and software engineering organisations at large, CMMI should not be an SDLC model but could rather be viewed, as a quality framework or approach which focusses on the process management principle. This principle ultimately states that a system's quality results from whether or not it was produced through quality processes (Chrissis et al. in Acton, Kourie & Watson 2014, sec. 2.8).

2.7.6 Six Sigma

Six Sigma attracts organisations from all sectors as it presents a comprehensive business strategy which is concerned with business improvement for operations excellence (Antony 2004, p. 305). It can be used for enterprise governance as well as the basis of a tactical improvement engine (Siviy & Penn 2005, p. 7). To achieve this, Six Sigma encourages organisations to focus on the reduction of variance while attentively paying attention to quality assurance (Anderson 2003, chap. 1).

Underpinned by the premise that reliability in the long run is determined by higher design margins, it posits that if such a condition is met, then production output will be able to withstand stress usage failures (Head 1994, p. 40). To achieve this, it additionally promotes the usage of statistical and non-statistical tools derived from previous business improvements initiatives (Siviy & Penn 2005, p. 8).

While this model was initially intended to improve manufacturing processes, most software engineering organisations that rely on it nowadays adhere to the “third wave” of Six Sigma implementation which has proven itself as a complement to previous models (Siviy & Penn 2005, p. 7). Six Sigma was at one point considered so effective that it helped the Motorola Group win recognition for its contribution to improving quality practices in the late 80s (Head 1994, p. 40). Consequently, organisations such as Motorola’s Global Software Group and many others have used Six Sigma or variations of it to complement the methods applied in the software CMM (Oster 2004, p. 68).

Although difficulties encountered by organisations are usually influenced by industry environment variables they need to consider, remaining competitive in a digital market place seems to transcend domain specific border lines (Schwartz & Amaba 2017, p. 427). Before anything else, firms are increasingly required to have more organisational agility in that markets expect them to quickly and efficiently adapt to environmental change. This implies organisational improvements in terms of balance, flexibility and coordination to better meet market needs (Boer & Sileno 2013, p. 5). For most enterprises including software engineering firms, this suggests rigorous and constant improvements in quality delivery of products and services to ensure their survival.

Considering the common goal of growth as well as quality delivery of objectives across industries, one could discuss restrictions to quality models both generically and in the software engineering world. The next section attempts this with a discussion on limitations to quality models previously discussed.

2.7.7 Caveat to traditional quality models

The models discussed above aim to provide organisations the ability to gain insight and a degree of control over the quality of their products. This is done for the most part by relying on models which use metrics used to identify low points in terms of quality and how to improve them. In studies such as the current thesis, there is a defendable temptation to rely on industry prescribed models which seem to have made their marks across the board and are generally recognized as satisfactory.

Nonetheless, a big issue with some of these models is that they can often be perceived by some smaller businesses to be either too broad or too narrow for specific uses in certain cases. Some studies have shown that businesses have at times considered that heavy footprint methods such as CMMI are preferable only when delivery timeliness was not paramount (Kalermo & Rissanen 2002, p. 83).

With older approaches, such as Hyatt and Rosenberg's quality model for example other issues need to be considered. In effect, despite the relative attractiveness of leveraging hard and tangible project metrics for better risk assessment to improve quality, the broadening of evaluation scope to processes and resource can still be considered too rigid. This model for instance does not cater for relevant data which could be qualitative in nature such as user's feedback, subject matter experts opinions or customer input (Hyatt & Rosenberg in Trendowicz & Punter 2003, pt. 3).

As a further illustration with more recent models, the CMM framework is perhaps another example of approaches which do not address all issues in software projects.

In effect, CMMI could be interpreted as only leveraging capability levels (CLs) to in fact address the issue of improving a firm's maturity level (ML); through these lenses, the model can be criticized as actually solely being uni-dimensional by only addressing this narrow aspect of organisational maturity (Ghanim 2016, p. 8). It does not clearly separate the maturity levels from other dimensions such as the detailed set of techniques, responsibilities, methodologies which need to be thoroughly explicated for businesses to evolve in the current competitive knowledge economy (*ibid.*, 2016).

In the case of Six Sigma, one could question its applicability in settings which require a broader perspective than narrow statistical analysis. Motorola advocated this approach to the issue of quality by concluding that setting a tolerance for defects under 3.4 per million units was a premise conductive to fostering higher levels of quality. This was argued to be effective in the automotive industry and even perhaps in some service-oriented sectors (Chassin 1998, p. 567). This also inspired Motorola itself and many other organisations into applying this standard deviation centric approach as the basis for quality strategies in other aspects of their businesses. Yet, some studies have showed that applied to some parts of the healthcare sector for example, Six Sigma implemented as per Motorola's initial suggestions could actually result in increased death rates for patients (Chassin 1998, p. 569). If one considers the importance of software systems in modern healthcare practices, the dangers of blindly applying Six Sigma hence become apparent because the nature of business processes themselves does not always make for easy or logical comparison across organisational settings (Antony 2004, p. 304). Furthermore, in software engineering settings, IS research suggests that practically attempting to measure a million scenarios in which one could introduce defects in any give software product is not a realistic expectation (Whittaker & Voas 2002, p. 31).

Other formal methods also presented challenges to organisations and their software teams in their attempts to satisfy their quality expectations. In effect, even older models such as Cleanroom were challenged because of the perception they were stuck in even older notions dating from the 1960s (Beizer 1997, p. 15).

A problem with Cleanroom methodologies is that they did not initially cater in a rigorous manner for unit testing (Head 1994, p. 41). In practice nowadays, teams which have understood the actual value of testing and where approaches such as Test-Driven Development (TDD) can be leveraged; no longer require to be swayed by models such as Cleanroom Software Engineering and many of its obsolete aspects if one considers for instance the current options available to organisations and engineering teams in terms of novel testing frameworks.

Despite criticism discussed earlier, for modern software organisations, the models discussed previously all influenced or contributed in improving restrictive older waterfall-like models traditionally championed by large multinationals such as Microsoft, IBM etc.

Today, engineering project approaches have evolved from these older methodologies to what would eventually be known as agile software development. In the next chapter, these agile ways of approaching software engineering and their origin are presented.

2.8 Chapter Two summary

Chapter Two served as the first layer of our literature review. It aimed at providing insight into what software quality has historically entailed. From a conceptual point of view, grasping the unique nature of software contributed to observe the need for formalizing software engineering as a discipline.

From this point on, a discussion was conducted to describe how this formalization of engineering practices evolved throughout time. Quality management and resulting quality models were explored to understand how organizations have attempted to deal with delivering their software projects. Caveats to these traditional models were nonetheless presented to understand why engineering teams have started to shift towards more agile approaches to software projects. In this light, Chapter Three serves as the second layer of our literature review by following. The next chapter of this thesis hence serves at discussing how agility as emerged as key requirement for the development of successful software projects in more recent times.

3. CHAPTER THREE: AGILE DEVELOPMENT APPROACHES

Chapter Three picks up from the first of half of this thesis' literature review. This continued literature exploration discusses how agile methodologies came to be an attractive evolution for software engineering projects execution. The chapter goes on to present customization options in terms of process and method engineering. It then deals with the tailoring of practices in agile settings. A conclusion finally follows with a presentation of disruptive agile technologies available to enhance the management software projects.

3.1 The origins of agile methods

Traditional SDLC approaches to project delivery were heavily reliant on requirements gathering and its accuracy. Some of these approaches such as the Waterfall Model were established to help organisations reduce the time needed to implement, build, test and ship software products to end-users (Bernstein & Yuhas 2005, p. 16).

Despite their initial allure to most software project engineering teams, these methodologies were not without drawbacks. For example, a big problem for project teams included the fact that estimations could often be ill informed and thus inaccurate for key project milestones. In effect, expecting customers to know, explicit and understand in detail every specific need and expectations during project inception can lead to false assumptions as requirements tend to change. In addition to this, many software projects have often failed when requirements have been gathered inefficiently or not precisely enough (Leffingwell 2010, p. 6).

These limitations and historical failures of waterfall models inspired the advent of an engineering philosophy initiated more than 75 years ago: Iterative and Incremental Design and Development (IIDD) which were basically a collection of structural software development models aimed at reducing the time it took to plan projects, build software artefacts, test them with users, and deliver them to customers in an efficient as well as cost effective manner for organisations (Deek, McHugh & Eljabiri 2005, p. 16).

Iterative Incremental Design and Development approaches where actually born from quality research conducted in the 1930s by Walter Shewhart at Bell Labs (Shewart in Larman & Basili 2003, p. 47). Later on in the 1960s, industries started modelling their processes iteratively because research had shown that certain behavioural aspects of systems built only appeared post implementation (Zurcher & Randell in Mckenna & Whitty 2013, sec. 2).

Throughout the 1990s, IIDD was commonly adopted by software engineers in the form of rapid application development (RAD), as well as the rational unified process (RUP) (Glazer et al. 2008, p.4). IIDD suggested a framework aimed at increasing engineering velocity by adopting an experimental approach to innovation discovery in the form a spiral experimental model (Leffingwell 2010, p. 10). Table 3-1 provides a snapshot of IIDD models (*ibid.* 2013):

Table 3-1: IDD Models

SPIRAL MODEL	RAD	RUP
Requirements gathering, and the validation of such requirements are the vital trigger to engineering projects. Although it is very similar to traditional waterfall sequential approaches, it innovated in its discovery-based paradigm regarding requirements gathering.	*It advocates for the iterative development of increasingly functional prototypes. Its main aim has traditionally been to accelerate availability of working software and is hence a regarded a key source of inspiration for modern agile practices.*	*Ingrained in the spiral model approach, it was meant to be used essentially in large scale projects where scalability, and robustness are essential. It recognises the need for flexibility in the management of activities as certain tasks need to be repeatable.*

Right before the start of 2000, methodologies geared towards more agility were introduced in engineering projects to suggest the adoption of predictable lightweight, efficient and scientific approaches to develop software (Beck in Mnkandla 2008, p. 41). Mnkandla describes the connotations implied from above as:

Table 3-2: Agile Methodologies features adapted from (Mnkandla 2008, p.41)

LIGHTWEIGHT	*This implies reducing software tasks to the essential and focusing on the delivery velocity,*
EFFICIENT	*This implies the delivery of solutions to targeted audiences with minimal overhead on the software project team.*
LOW-RISK	*This implies that Project Managers and their teams trade on the practical lines of what is feasible and leaving the unknown until it is unavoidable.*
PREDICTABLE	*This implies relying on common agile practices.*
SCIENTIFIC	*This implies basic methodology execution on formal scientific principles.*

It is against this backdrop with the proliferation of IIDD methods and the need for increasing agile ways of doing software development that field authorities met in the Wasatch Mountains of Utah in February 2001 (Glaz*er* et al. 2008, p. 4).

From this meeting, emerged an *Agile Alliance* on one hand; and on the other the birth of the so-called "*Agile Manifesto*". This manifesto essentially aimed to value core aspects of software engineering projects in so-called agile settings (Glazer et al. 2008, p.16). These values include:

- An emphasis on individuals and their interactions as opposed to specific tools or processes.
- An emphasis on functioning systems as opposed to large sets of documentation.
- The importance of prioritising cooperation with customers as opposed to monetary disputes over service agreements.
- Adapting to changing requirements in project scopes rather than following set-in-stone plans.

Ultimately, the Agile Manifesto encourages principles revolving around increased development speed, collaboration and flexibility in terms of running software engineering projects (Lytvynova 2018, para. 2). The current thesis considers agile described methodologies which foster and encourage iterative development processes in an incremental manner as described by local software engineering literature (Lindvall et al. in Mnkandla 2008, p. 42).

3.1.1 Advantages of the agile philosophy

Modern software organisations increasingly require agility in their approaches to software development projects (Gaoussou & Mnkandla 2018, pt. 1). This has been observed to provide organisations with increased flexibility and efficiency when facing continuously evolving requirements (Shelly & Rosenblatt 2012, p. 149). When adopting agile methodologies and techniques, the outcomes of iterations are expected to be completed and passed along after review, testing and acceptance by users (Scheid 2013, para. 7). Common perks which are often cited for the adoption of agile approaches include the fact that they are adaptable to unique projects and that they can be implemented into almost any setting where the traditional waterfall model's limitations are no longer tenable (*ibid.* 2013). When firms opt to utilize adaptive agile methodologies in engineering projects, they typically proceed to structure development processes around iterations which are focussed on delivering specific functionalities (Shelly & Rosenblatt 2012, p. 21). Furthermore, the incremental approach to engineering which the agile philosophy encourages has been found to enable the reduction of overhead costs, facilitate customer feedback and increase the delivery of end products to users in a more timely and efficient manner (Svorstøl 2017, p. 4).

As previous IS research has shown, when organisations migrate to agile approaches of executing their engineering projects, the potential to foster better conditions for the delivery of higher quality products increases (Qumer & Henderson-Sellers 2008, p. 280). Recent studies have for instance shown that in comparison to traditional waterfall like approaches, in agile settings, teams seem to show increased satisfaction in regards to their overall productivity as well as end product quality delivery (Kassab & Defranco 2018, p. 118). In recent times, agile approaches to software engineering projects have increasingly attracted both small and larger enterprises because of noticeable gains they provide in terms of timely delivery, efficiency and product quality improvements (Ambler & Lines 2012, p. 1). Recent industry surveys amongst professionals have for instance indicated various advantages of agile practices in organisations; these include an increase in customer involvement, improved collaboration across multi-functional teams, adaptability in the face of scope changes, parallel development and testing activities, reduced time to market delays as well as performance and overall product quality improvements (Parker 2019, para. 20).

Often associated with iterative development, simplicity, face-to-face communication and customer focused approach, agility presents an attractive paradigm for modern firms (Kingdon 2018, para. 1). It could hence be deemed pertinent to examine agile practices in further detail when one ambitions to study modern approaches to improve current software engineering projects.

In practice, the agile conceptualization of software development tasks project leads with the responsibility of establishing and allocating process activities (Perera, Bandara & Perera 2016, p. 282). Furthermore, there exists an immense diversity in the type of software processes organisations can leverage to build products intended for commercialisation to customers (Di Nitto *et al.* 2016, p. 13). This explains in part why there are so many different flavours of practical agile implementations such as Feature and Test-driven development in addition to Scrum etc. (*ibid.* 2016).

This non-exhaustive and ever evolving list of implementations of the agile philosophy presents researchers in the field with an immense amount of literature to review, dissect, and analyse.

In this light, the next section of literature review attempts to position itself within the South African context realities.

3.1.2 Agile software development in the South African context

As its international counterparts, the South African software engineering sector has embraced Agile Software Development endeavours. Locally, Agile is expected and has been shown to decrease IT software projects failure rates (Joseph, Marnewick & Jan Santana 2016, p. 338). Local academia seems to indicate that the most prolific agile methods used in South Africa are Scrum and Extreme Programming (*ibid.* 2016). Additionally, this research‘s case study is conducted at a local start-up which uses both Scrum and Kanban. The next sections hence proceed to introduce these three agile modern approaches to software development.

3.1.3 An overview of Extreme Programming, Kanban and Scrum

3.1.3.1 Extreme Programming

Extreme Programming (XP) appeared in 1996 as one of several popular agile processes to focus on delivering software as users required it (Wells 2013, para. 2). This methodology proposes an evolutionary approach to software projects focused on incremental processes, automated testing as well as rapid deliveries (Hoffer, George & Valacich 2016, p. 19).

XP advocates for short development cycles called releases to improve productivity (Mwansa 2015, p. 30). It proposes a disciplined engineering approach which is centred around the delivery of functioning products to customers in a timely fashion (Hunt 2006, p. 16). Furthermore, XP also advocates for effective and intelligent communication between various project stakeholders including management and engineering staff with a common goal to ensure project success. By leveraging XP, high velocity and efficient project teams typically ambition to deliver quality software products with as little red-tape and within the most appropriate time, resources and cost constraints as possible. Extreme Programming can also often be associated with a set of values and principles in most modern agile software engineering settings (Lytvynova 2018, pt. 2):

- Respect in the sense where all project stakeholders work in collaborative effort to achieve a common goal.
- Accountability in the sense where high locus of control is expected from engineers with a key emphasis on honesty and objective self-reflection as well as the willingness to accept new challenges.
- Simplicity in engineering activities. By encouraging developers to write simpler testable code units, increased product value is expected with savings in terms of efforts and time costs.
- Effective communication streams to ensure all project stakeholders work together throughout project lifecycles.
- Continuous Feedback loops to foster a culture of continuous rational evaluation and improvements in terms of product quality.

Another attractive aspect of XP is that it encourages practices such as pair-programming which can be introduced to novice engineers. Such approaches to executing development tasks have been shown to improve collaboration between engineers while producing better overall software artefacts (McDowell *et al.*, 2004, p. 37). In most settings, pair-programming endeavours have been shown to fit well within mature extreme programming frameworks (Bryant, Boulay and Romero, 2006, sec. 3).

Moreover, XP also emphasises the adoption of frequent cycles to improve development projects and promotes the identification of four core practice areas (PMI & AgileAlliance 2017, pt. Annex A3). Table 3-3 summarises these XP areas:

Table 3-3: XP practice areas and activities (PMI and AgileAlliance 2017, pt. Annex A3)

XP PRACTICE AREA	RELATED ACTIVITIES
Organisational	*Teams are expected to collaborate in informative workspaces while facilitating customer involvement during project lifecycles. Team members are expected to work at a very high but sustainable pace while team leaders must ensure continuity in any circumstances.*
Planning	*The team is expected to elaborate descriptive user stories and plan cycles weekly and/or quarterly. Daily stand-ups are also used to make sure all members are informed of current issues and update task status accordingly.*
Technical	*Methods such as pair and test-first programming are implemented. Engineers also need to understand and apply incremental designs to make refactoring efficient and effective.*
Integration	*Engineering staff ensures code bases are integrated and stable. The principles of incremental deployment, unit and integrated testing as well as continuous integration are implemented.*

XP additionally assists agile teams in overcoming the limitations of traditional SDLC in terms of delivery time, flexibility and adaptability (Sharma & Hasteer 2016, p. 1). XP is mostly used as a simple and effective methodology for small agile team (Beck in Haryono 2015, p. 28). According to Haryono (2015) its fundamentals include:

- The understanding that decisions influenced by business requirements and those made by project stakeholders have different origins.
- The importance of developing unit test cases prior to implementing functional logic and continuously testing against these.
- Performing rigorous integration testing.

- Producing all software in pairs by making to engineers collaborate on one screen.
- Initiating projects with simple, flexible and adaptable designs.
- Pushing light functions as soon as possible.

Figure 3.1 illustrates an example XP model which puts the emphasis on the iterative nature of agile.

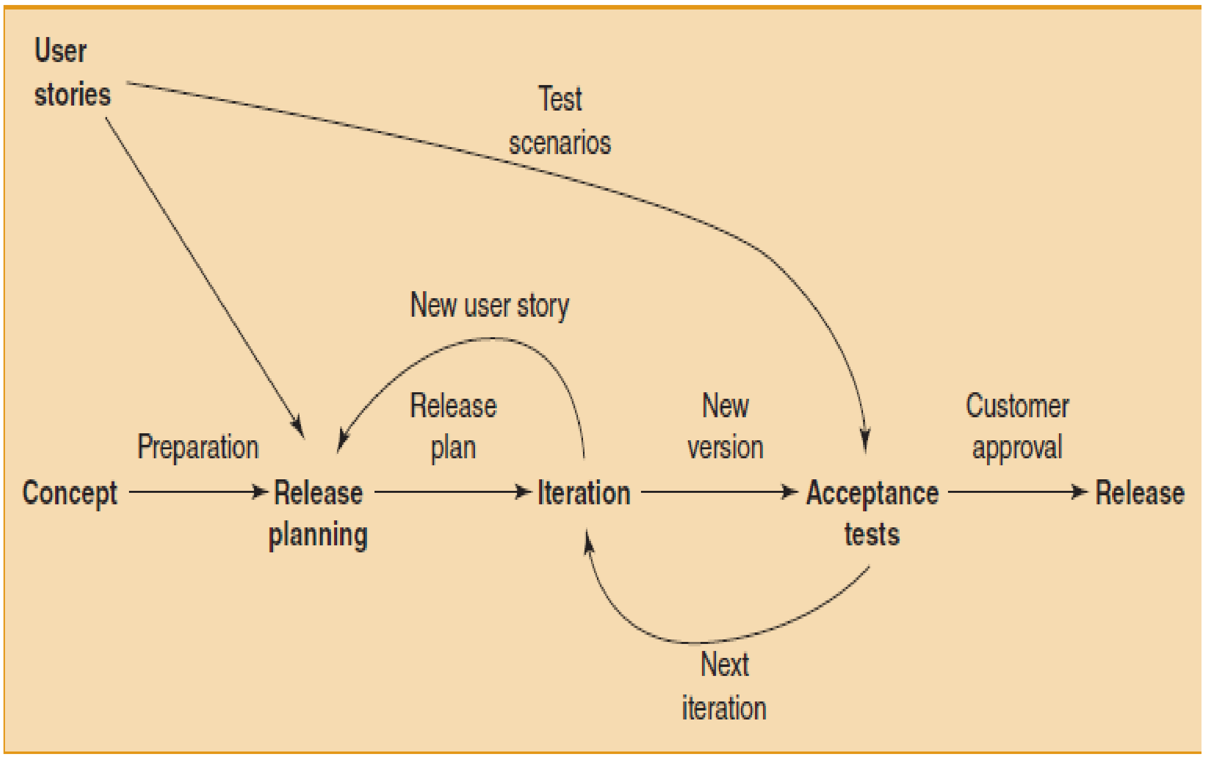


Figure 3.1 Simplified XP model (Shelly & Rosenblatt 2012, figs 11–7)

3.1.3.2 Scrum

Agile methodologies departed from traditional waterfall inspired models by providing a novel approach to project development (Cooper 2016, p. 24). Using the analogy of *"Scrum"* in rugby where a match is rebooted after every infraction, agile project stakeholders leverage short meetings to strategize and plan action items (*ibid.* 2016). Scrum teams typically feature stakeholders such as a product owner, a development team, managers and users amongst others (Shelly & Rosenblatt 2012, p. 148).

Scrum is such a popular framework for implementing the agile paradigm to software development that it is often used as a synonym to the term agile itself. It considers that software engineering projects are unpredictably complex and one needs to take account of this at inception (Mahnic & Drnovscek 2005, sec. 2).

The basic premise behind scrum is that the set of variables which influence software projects are not predictable and that imposed rules would not always be appropriate (Mnkandla 2008, p. 62). It is hence pertinent to adopt a methodology which spells out the best practices to manage such projects (*ibid.* 2008).

In practice Scrum distinguishes itself because of its insistence on committing to short units of work in increments with high velocity which help teams rapidly advance towards their stated goals (Yoshida 2018a, pt. 6).

Scrum expects products to be built in a series of fixed-length iterations called sprints which provide projects teams with a rhythm and framework to deliver software on regular cadences (Radigan 2017b, para. 2). These sprints in question are structured by four ceremonies identified as follows (*ibid.* 2017):

- ***Sprint planning*** in which stakeholders organize a meeting to determine what to complete in the coming sprint.
- ***Daily stand-up*** which entails a daily fifteen minutes meeting for the software team members to update each other.
- ***Sprint demo*** which entails a sharing meeting where the team shows deliverables achieved.
- ***Sprint retrospective*** which entails a review of positive and negative items to improve future sprints.

Agile organisations and teams use pull models where they retrieve a certain amount of work off the backlog and commit to completing this work during specific sprints; this is a very effective way of maintaining quality and ensuring optimum performance in the long term (Radigan 2017b, p. 1).

Mwansa (2015) summarizes Scrum as an iterative and incremental project management framework. This framework takes into account the evolving nature of customer requirements and therefore aims to encourage flexibility and adaptability from software project teams (Mwansa 2015, p. 30). This declination of the agile philosophy also encourages the presence of end-users throughout the development lifecycles because requirements are understood to be evolving (Lopez-Martinez *et al.* 2016, p. 142). Agile trained project leaders promote Scrum to organisations by describing it as a management framework which facilitate incremental and agile development (Hutterman 2012, p. 23). Companies tend to find this attractive as Scrum fosters an environment where business stakeholders and technical staff collaborate at regular time intervals (*ibid.* 2012). Furthermore, as Scrum is a concept closely related to agility, it is at times used as a synonym for agile practice. By itself, although it does not force restrictive guidelines on the rituals teams should perform during projects, it is often suggested that organisations combine a scrum inspired framework with specific practices geared towards compatibility within their environment (Ambler & Lines 2012, p. 44).

Another element which differentiates Scrum teams from teams found in traditional waterfall models is their composition. There are three essential roles in a scrum team which include engineering teams, scrum masters and product owners. Table 3-4 describes these roles in more detail (Radigan 2017b, p. 1):

Table 3-4: Scrum team composition

THE PRODUCT OWNER	*Product owners focus on understanding business requirements before prioritizing the items to be completed by the engineering team. They manage product backlog; ensure smooth communications between business and engineering teams; give clear guidance on outstanding features; decide when the products are shipped. A key aspect to remember about such stakeholders is that they are not meant to replace project managers. They are usually individuals as to have a multitude of product owners could result in the development team receiving mixed signals.*
THE SCRUM MASTER	*Scrum masters are tasked with coaching development team, product owners and business stakeholders on scrum processes while attempting to fine-tune their practice of such processes. They can be deemed as facilitators-in-chief as they manage project resources requirements, lead stand-up meetings, review all sprints and evaluate the retrospective phase.*

THE SCRUM TEAM	*Well organized and efficient scrum team are usually closed, co-located and rarely exceed 7 members. The members of such teams usually offer projects differing skill sets, and continuously assist in cross-training members as to avoid an individual being a bottleneck. They need to be able to plan sprints, forecast their expected delivery rates over each iteration by using their passed velocity record as guides.*

3.1.3.3 Kanban

Kanban can be described as an agile development methodology inspired by Toyota in the late 1940's (Radigan 2017a, para. 2). It was derived by the car manufacturer's JIT (Just-In-Time) philosophy to deliver final products in a timely fashion (Christian 2016, para. 3).

The key aspects one could retain when discussing Kanban is that it promotes transparency, efficient communication and clarity in terms of capacity and tasks allocation. In Kanban projects, items are visualized by putting cards onto a task board (Beck et al. in Nakazawa & Tanaka 2016, p. 1). This way of managing software projects has a low barrier to entry for teams newly adopting lean and agile principles and aims to drive improvement by facilitating the visualization of work items represented by cards on task boards for example (Kelly 2017, para. 2). Cards on these task boards will often include technical details valuable to assignees and allow Kanban teams to view tasks status throughout the project at any time. This facilitates auditing by enabling timely identification of stumbling blocks while ensuring increased focus (Radigan 2017a, para.10). The Kanban approach also attempts to improve efficiency by placing constraints on the number of items (Work In Progress) which project teams can tackle simultaneously, this is argued to improve focus and avoid situations where teams are wasting time by dispersing their efforts (Majowska 2018, para. 1).

Kanban tries to focus on facilitating continuous development and promote continuous delivery by managing work requests in parallel and in a seamless fashion (Rehkopf 2018, para. 4). Typical Kanban projects are highlighted with a flowing cadence, a continuous release approach to deliveries, little emphasis on existing rules and a constant adaptability to change (*ibid*. 2018). Another attractive aspect of Kanban is that it is simple to adopt for organisations with a relative low level of knowledge expertise about agile and lean paradigms. Additionally, previous research work as shown that many organisations are attracted to Kanban because it is perceived to help reduce Work In Progress (WIP), facilitate increased visibility and accountability for delivery leads while improving overall development flow (Ahmad, Markkula & Oivo in Helen & Tracy 2016, p. 165).

Furthermore, in common agile settings, incidents occur and much of the work is often more unplanned than not. When new work requirements are introduced at high velocity, teams cannot afford to spend long amount of time in trying to make sense of backlogs or wait for upcoming sprint planning sessions (Buchanan 2018, para. 4). This is amongst others a core motivator as to why many agile teams opt for Kanban in the running of their projects.

3.1.4 Limitations to current agile approaches and prospects

Agile implementations can be painful endeavours when there is not a willingness from within organisations to accept the need for change. In effect certain studies have shown for instance that factors such as people and the organisational culture are a huge reason for resistance to change and that in itself is one of main challenges to successful agile adoption (Lopez-Martinez *et al.* 2016 p. 146). During attempts to adopt practices such as XP for instance, leadership is often met by reluctance from engineers. This is because they often believe that pairing with colleagues would lead to a loss of efficiency in their programming habits and force intense social interactions they feel uneasy with (William et al. in Kalermo & Rissanen 2002, p. 100).

On a more practical level, during some engineering projects, teams tend to discover that high velocity new work requests, continuous change and continuously updated requirements can immensely hamper agile implementation. In effect, many of these issues which can cause project instability do not coincide well with a restrictive Scrum or Kanban adoption for example (Daly 2018, para. 8). Consequently, many projects need to blend compatible practices from various agile approaches rather than use prescriptive guidelines based on a specific methodology.

Switching from predictive approaches to more adaptive models can present organisations which are ill prepared some challenges. In effect, it has been argued that too much emphasis on the rapid iterative aspect and quick deliveries in agile shifts focus away from precision and can yield an atmosphere which fosters overall indiscipline (Shelly & Rosenblatt 2012, p. 523). Additionally, doubts have also been emitted about agile efficiency in the context of large-scale projects because of the perceived absence of clearly explicated user requirements during early stages of project inception (ibid., 2012). The migration from typical SDLC approaches to new methods can also yield other unintended situations.

To evolve their practices in the context of software project execution, while proceeding to a transition from traditional waterfall models towards more agile practices, project engineering teams are at times tempted to completely forgo any sort of planning (Monson 2019, para. 29). This is very problematic and is often motivated by the desire to leverage a new so called agile midframe as a remedy to bloated documentation, confusing Gant Charts or over ambitious project plans (ibid., 2019).

What happens as a result of this is that organisation end up in a sort of hybrid model which has all the disadvantages of waterfall processes accentuated by a bad understanding of agile. This can result in unmanageable software architectures which exponentially grow complex:

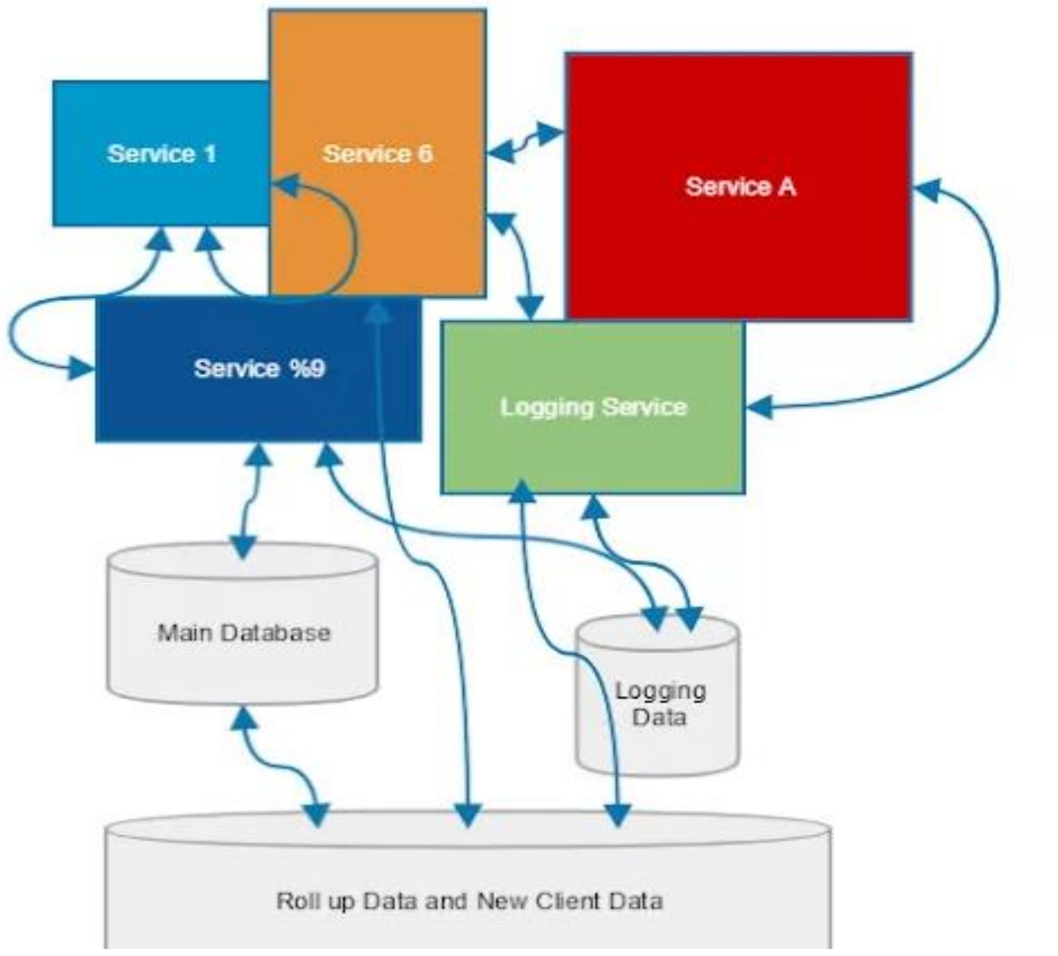


Figure 3.2 Software architecture example derived from bad agile implementation (Monson 2019, fig. 3)

Another issue in enterprises is that most organisations find keeping small agile teams focused on the same long-term objectives very difficult. Although this is not a problem exclusive to agile oriented engineering projects, maintaining coherence in terms of unity, logic and consistency for instance is a daunting task *(Glazer et al.* 2008, p. 20).

In practical agile implementations such as Kanban or Extreme Programming for instance, specific problems can also be encountered in software engineering projects. XP for example was initially intended to be adopted by small teams. In the real world, scaling up of practices such as pair programming for larger organisations is not a simple or smooth process (Kalermo & Rissanen 2002, p. 103).

In organisations which attempt to run projects in Scrum settings for instance, post mortem audits have often shown that writing meaningful user stories can prove to be very complicated because of the perceived lack of clarity associated with agile adoption attempts; furthermore, to advance positively in their agile journey, teams need levels of adaptation and transparency which can be lacking throughout many companies (Lopez-Martinez *et al.* 2016, p. 146). Additionally, after the implementation of Scrum, some teams can encounter difficulties which can lead to project failure. These failures can be caused by a few reasons as discussed in table 3-5:

Table 3-5 Scrum Failure Factors (Adapted from Liyanarachchi 2019, p. 1)

SCRUM FAILURE FACTORS	**DESCRIPTION**
Gaps in comprehension of scrum principles.	Although in most environments, the entire project team would have gone through some type of training or introduction to scrum, roles and expectations need to be explicated to all team members. Locus of control lack is problematic and ends up resulting in failures at times, so it is vital the Scrum master endorses the responsibility of coach team members and the onboarding of new colleagues. This can assist in ensuring all stakeholders have a common understanding of the agile practices and processes necessary to a successful Scrum framework implementation

Inappropriate application of scrum without considering specific project needs.	Projects are at risk when Scrum is adopted without understanding the nature of team dynamics. When team members do not have shared skills or experience, leadership needs to shy away from simply putting together an ensemble of incompatible subject matter experts.
Mistaking scrum for a monitoring framework.	Leadership needs to use daily stand-up meetings as soft debugging session where engineers collaborate, advise and communicate to deal with issues. These sessions are wasted when only use to monitor progress without tangible inputs from the team in fixing issues.
Rigid top down management of project teams.	Scrum projects lose their purpose if management does not enable self-organisation. Individuals need to be able to leverage cross-functional competencies from colleagues. Without this, project team members are at risk of losing focus and motivation, and simply revert to negative attitude in the face of perceived micromanagement.

In the same light, Glazer et al. (2008) further advance that because of certain prescribed agile activities, complex software engineering projects have difficulty scaling well. The influencing factors include:

- Clarity for teams in terms of understanding business requirements.
- Product and processes alignment amongst teams.
- Managing validation and verification strategies for the solution.
- Coordinating risk management.
- Fostering a common team vision and the understanding of objectives by all.

Another problem which teams can encounter when trying to implement agile practices is that of quality. In practice, it has been observed for example that agile software engineering projects have often lacked suitable techniques to address quality requirements (Inayat et al. in Knauss *et al.* 2017, p. 427). Consequently while agile implementations often coincide with noticeable improvements in terms of quality, traditionally speaking, at inception, Scrum and Kanban never explicitly stated that organisations should focus on improving their quality per say (Kelly 2017, para. 4). Additional impediments to agile approaches implementations in organisations include a lack of management support and general resistance to change (Glazer *et al.* 2008, p. 26). The factors mentioned above are not the only issues organisations face when trying to implement an agile philosophy.

For example, one of the basic pillars of most successful agile implementations is that of communication. In effect, it is a requirement that aside from technical aptitude, project members be highly skilled at communicating effectively (Shelly & Rosenblatt 2012, p. 21). Also, a key requirement of effective agile practices is to as much as possible encourage direct face-to-face communication between project stakeholders (Kingdon 2018, para. 6). Additionally, the lack of emphasis on documenting agile projects can increase risk factors and team leaders need to understand that because of the continuously evolving nature of requirements, scope changes are not unlikely, so the project team needs to adapt accordingly (Shelly & Rosenblatt 2012, p. 21). Coupled to this, some practitioners have also identified a tendency in certain projects to approach scrum planning without a clear understanding of who the actual target audience is (Yoshida 2018, para. 1). This can for example yield a lack of clarity in the motivations for adding certain items in sprint backlogs to all stakeholders and hence overload teams with unnecessary workloads.

In the foreword to a Scaled Agile Inc. white paper, the authors ‘ main argument is that when organisational change is not in phase with the global business environment, firms tend to go on a downward spiral (Scaled Agile 2016, sec. Foreword). In effect, while agile development has addressed in large parts the need for responsive development, it was developed for small teams and scales difficultly in large enterprise systems (*ibid.* 2016). Additionally, in the context of satisfying customers’ desire for fast and continuous innovation, agile development only fills the gap between Line of Business and Engineering teams (Bakal 2012, p. 4). This situation often leads to increased misalignments between project aspirations and practical engineering challenges, vacuity between objectives and deadlines as well as mismatches between expected performance and delivered project features (Raam 2017, para. 4).

To bridge the gap between Engineering and Operations teams, modern organisations are increasingly looking to “*DevOps”* ("development" and "operations") as an agile enhancer.

3.1.5 DevOps necessity in the agile paradigm

As Agile projects typically try to ship features at the end of each sprint, this tends to mechanically incur high deployment rates for IT Operations (Kim 2015, p. 5). A consequence is that while agile development has positively influenced businesses' perceptions of software teams, it has conversely negatively affected the perception of IT Operations performance (*ibid.* 2015). This has led to the emergence of "DevOps" as new engineering method to address some of the agile philosophy's practical implementation challenges.

For many software teams or organisations, operations and development functions have conventionally been diversely considered and appreciated. A software engineer has typically been understood to be the builder of code or software while operation staff has usually been tasked with the responsibility of production or operational support. A direct consequence of this has been the disconnect between members of these teams. In practical industry settings for instance, application internals as well as detailed software architecture are not always well understood by operations staff (Di Nitto *et al.* 2016, p. 12). Conversely, software engineering teams are not always well in tune with infrastructure and critical environment variables such as platform dependencies, network limitations etc. (*ibid.* 2016). Another problem which engineers in software projects have traditionally struggled with is to ensure standardization of development, testing and production environments. To ensure consistency across these environmental domains, organisations have in the past mostly relied on change management processes (Poojary 2015, para. 5). In modern times, DevOps helps teams conduct shorter build, test and deployment cycles by automating many engineering processes (*ibid.* 2015). Industry use cases at technologies giants such as Flickr and Netflix attest to a growing interest in integrating development and operations activities (Bang *et al.* 2013, p. 61).

Recent research as argued that when applied correctly, DevOps can extend the agility of software engineering teams to the operations function and that it could help achieve high levels of efficiency during the entirety of planning and deployment lifecycles (Kamuto & Langerman 2018, p. 48).

The term "DevOps" refers to an innovative paradigm which emphasises increased collaboration, communication and integration between software engineers an IT operation professionals (Happiest-Minds 2014, p. 3). It is a contraction of the terms "development" and "operations" and aims to enhance agility by providing practical solutions to challenges in areas such as environment configuration, continuous integration as well as tasks revolving around monitoring and performance improvement (Gartner 2018, para. 9). One could also describe this approach as a set of techniques which facilitates enhanced communication, collaboration between software and operations engineers. The ambition of DevOps teams is to produce quality software faster and more reliably at high throughput rates. To achieve such feats, organisations need to migrate away from traditional horizontal divisions of labour amongst disparate functional teams (Balalaie, Heydarnoori & Jamshidi 2016, p. 8). A couple of consequences of dividing work in such ways include an increase in delays experienced for change delivery, less flexibility in the assimilation of new team members and reduced comprehensibility of source code which is often heavily correlated to the nature and number of teams involved (*ibid*. 2016). An additional issue which can accentuate this situation is that in traditional development teams, internal challenges emerge because often, development tasks are executed in silos. This has a nefarious effect for organisations in that, in some cases, only specific staff gain expert knowledge in their respective domains without ever taking into account the organisational need in the long term for effective knowledge sharing (Lavallée & Robillard 2015, p. 686).

To address the bulk of problems mentioned above, DevOps advocates a more vertical approach which suggests the allocation of team members to smaller cross-functional teams comprised of competent poly-skilled and adaptable agile engineers. While companies can admittedly deliver many features and products while operations and software engineering teams work in silos, there is a risk involved where support staff might be testing products in non-realistic environments and this can lead to decreased user satisfaction (Raam 2017, para. 3).

Therefore, in the broader context of modern engineering teams, DevOps ambitions to apply the agile philosophy and agile precepts beyond development team by associating IT Operations engineers. A key motivator to highlight the need for this collaboration between traditionally distinct IT functions in the form of development and operations is to eliminate conflict by bridging the gap between software and operations engineers (Wahaballa *et al.* 2015, p. 211). Organisations can ultimately be very attracted to DevOps implementations for their projects as it advocates for faster response from teams in charge of IT infrastructure and the software platforms required by business and customers alike.

Conceptually, DevOps represents a model for application development in agile environments which requires deep collaboration between the development and IT operation functions. It emphasises the need for the agile manner in which innovative modern software is developed to align with the way in which it is delivered to live environments (Debois in Fitzgerald & Stol 2017, p. 3). Collaboration between operations engineering and software development team is hence deemed to be critical in DevOps environments. This collaboration is argued in the long run to improve overall communication and help teams identify customer pain points quicker so that effective and efficient solutions are implemented decisively to improve customer experience and satisfaction (Parzych 2017, pt. 2).

For organisations which aim to successfully implement DevOps and yield expected benefits from its adoption, its introduction adds more responsibilities on the shoulders of leadership staff; management is hence put under increased pressure to foster an environment which is compatible with enhanced automation capabilities and encourage better locus of control by facilitating self-organisation in teams (Kamuto & Langerman 2018, p. 49). A key underlying ambition of this approach is to improve alignment between tooling, processes and tools in modern engineering settings (Dawson 2017, para. 7).

Nowadays, DevOps is also considered as an absolute requirement for the development of quality products and the effective running of resilient systems supported by increased levels of adaptability as well as change delivery (Dyck, Penners & Lichter 2015, p. 3). It is increasingly attractive to all size organizations which aim to improve flexibility, enhance productivity gains and experience rises in quality delivery in the context of software engineering projects delivery (Sereda 2019, para. 23). Enterprises are eager to adopt it for improved speeds in terms of project execution, reduced costs and increased chances of delivering quality software (Bertham 2018, para. 1).

Another perspective DevOps could also be thought of, is that of a modern, practical implementation of techniques inherently compatible to the agile philosophy in general. The foundational theoretical blocks related to DevOps are presented in table 3-6 (Cois 2014, p.3):

Table 3-6: DevOps terminology (Cois 2014, p.3)

TERM	DEFINITION

DevOps	The term itself entails all that is necessary to take an idea (feature, code, documentation etc.) from inception through delivery to a customer in the most expedient and sustainable way possible. This model can be applied to the development lifecycle of anything with distinct development and operations components.
Continuous Integration	Continuous integration (CI) refers to the combination of development source code into a single application and then, typically running automated suite of tests of the resulting system. This integration process runs "continuously" by either polling source control systems at regular intervals or by having hooks setup which are triggered by code updates to the repository where it sits.
Continuous Delivery	Continuous delivery refers to the process of packaging, testing and storing an application unit in a continuous manner so it can be ready to be deployed into production in a timely manner. Continuous Delivery extends CI process to arrive at an application which has gone through rigorous compliance testing and is validated to be production ready.
Continuous Deployment	Continuous Deployment refers to automated deployment of applications to production environments. This typically supposes robust production validation mechanism and smooth rollback capacities.

Figure 3.3 illustrates the processes expected to be in place throughout teams which implement DevOps:

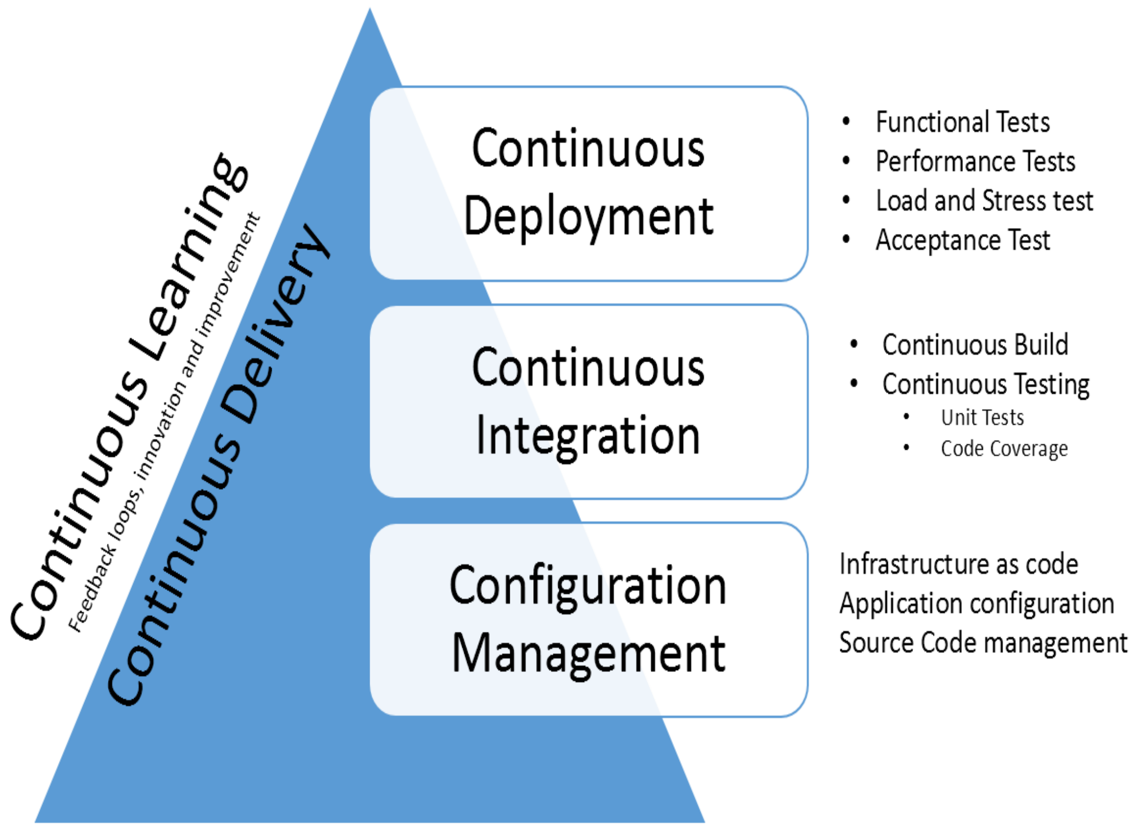


Figure 3.3: DevOps processes (Modi 2017, fig. 2)

The move towards agile methods for software development teams and the adoption of DevOps enabling practices increasingly suggests a shift away from rigid traditional methodologies and customize modern approaches to execute software projects. In this light, the next section discusses why modern organisations and software project teams need to progressively leverage more flexible processes

3.2 Adjusting engineering approaches

Software engineering advocates disciplined rigor in the utilisation of software processes in the running of projects. Understanding and managing software processes effectively is paramount to organisations which aim to improve productivity as well as product quality delivery (García *et al.* 2007, p. 2570).

In an engineering context, processes will typically preconize a set of well-known established practices in the aim of improving productivity while yielding quality software artefacts and end-products (Casare *et al.* 2016, p. 197). For example, many programmers find comfort in following specific formal guidelines in already well established methodologies (Karlsson & Wistrand 2006, p. 83).

Nonetheless, efficient software project teams understand that processes can typically not be applied to any random setting without at least some sort of process customisation being required (Kalus & Kuhrmann 2013, p. 1). It has hence been observed that software teams and organisations in particular need to continuously adapt their operating models and processes to meet internal and external evolving requirements (Xu & Ramesh 2008, p. 39). Consequently, it is also important that project leaders tailor their engineering processes considering the reputational and financial risks involved when running new projects (*ibid.* 2008). This is even more important for organisations which have certifications imperatives such as ISO or CMMI as in such cases, usage of industry accepted standards and processes is compulsory (Hurtado Alegría *et al.* 2011, p. 4).

Overall, when delivering new software products to customers, teams tend to attempt the customisation of methods to stabilize engineering projects and improve their own internal processes to deliver quality end-products to markets (Akbar *et al.* 2014, p. 1).

The next section proceeds to introduce different angles an organisation could leverage when attempting to adapt, customize and thus and improve processes in the running of its software projects. The rationale for discussing these options is related to the fact organisations rarely encounter one size fits all models randomly applicable to any setting.

In this section, engineering meta-modelling as well as method engineering are further explored. The subsequent section aims to discuss how these approaches are informed by a need for the contextualisation of engineering project execution to organisational specific needs and thus infers a need for the tailoring of processes in modern agile settings which are the main premise of this dissertation's use case.

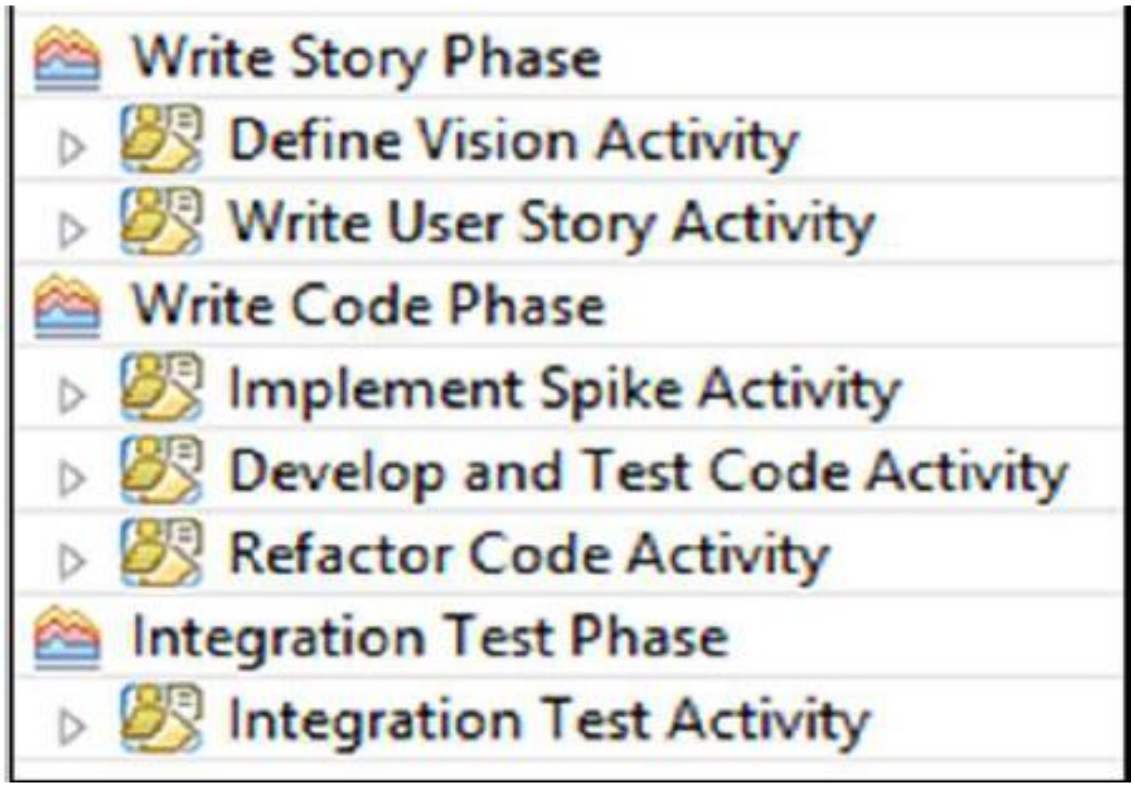
3.2.1 Software and System Process Engineering Meta Modelling

In the context of software engineering, organisations have a vested interest in formalising internal processes as efficiently as possible. In this light, meta-models such as the Software and System Process Engineering Meta-Model (SPEM) have proven to be quite useful (Akbar *et al.* 2014, p. 1).

Developed by the Object Management Group (OMG), the Software and System Process Engineering Meta-Model describes software and engineering processes as well as their components (Alajrami, Gallina & Romanovsky 2016, p. 2).

Its 2.0 version (SPEM2.0) was generically designed to accommodate a large array of engineering approaches and processes by abstracting domain-specific variables from its foundational structure (*ibid.* 2016). This notation has for instance previously been used to simplify processes relevant in agile scrum settings in the form simple stages such as (Beck in Akbar *et al.*, 2014, p. 197):

- *A Write Story Phase*
- *A Write Code Phase*
- *An Integration Test Phase.*



Write Story Phase
- Define Vision Activity
- Write User Story Activity

Write Code Phase
- Implement Spike Activity
- Develop and Test Code Activity
- Refactor Code Activity

Integration Test Phase
- Integration Test Activity

Figure 3.4: Agile Software Process Example (Akbar et al. 2014, fig. 1)

Models such as SPEM can be immensely useful to organisations because they help facilitate harmonisation and standardisation throughout engineering project lifecycles which in turn improves communication and harmony between project stakeholders. While this could hypothetically contribute to improve working conditions and potentially increase the chances of yielding quality products, it is not without potential limitation or risks.

In effect, intrusive and imposed standardisation does not guarantee immediate full user adoption. It also poses the risk of limiting flexibility by possibly enforcing one-size fits all solutions to very diverse environments (Harmsen, Brinkkemper & Oei 1994, p. 4). To address this potential pitfall of models, traditional engineering approaches to contextualise enterprise specific contexts could provide some relevant inspiration.

3.2.2 Method Engineering

In IS research, Method Engineering (ME) deals with all aspects related to the conceptualisation, development and customisation of new tools which support methods and techniques required for running adaptive software projects (Brinkkemper 1996, p. 276).

When addressing limitations and efficiency in the running of engineering endeavours, project leaders have understood that methodologies need to be contextualised to an organisation's realities (Ralyte *et al.* 2003, sec. 1). Considering this, method engineering emerged to suggest that teams initiate projects by leveraging components of existing methodologies as a foundation. In Method Engineering, these components are labelled as *method fragments* which are meant to be adapted to meet specific needs and assist in deriving, designing novel methodologies or approaches (Cossentino & Seidita 2005, p. 1).

Regarding software process improvement (SPI), Method Engineering has been shown to be useful when agile only focusses on the methodological aspects of projects. Previous research has for instance suggested that whereas agile methods typically mostly allows flexibility in regards to methodology, method engineering can be pertinent enabler to SPI when one leverages other meta-models such as the Open Process Framework (Henderson-Sellers & Serour 2005, p. 2).

To facilitate method engineering support in engineering teams, previous research has for example suggested SPEM inspired frameworks to combine method design, configuration as well as implementation for a better customization of software project execution (Cervera *et al.* 2011, p. 141).

Nonetheless, some studies have advanced that agile approaches are not necessarily incompatible with method engineering principles in the sense where they revolve around a core lean value which is to emphasise the adoption of only what one requires from a given methodology (Highsmith in Fitzgerald, Hartnett & Conboy 2006, p. 199). Drawing from this customizable aspect of ME, one could apply the same approach to attempt the tailoring of agile practices in organisational settings.

3.2.3 Tailoring agile processes

While organisations can find solace in implementing industry tried and tested methodologies and approaches to their engineering projects, teams running these projects additionally require flexibility in adapting plans and methods on the fly (Karlsson & Wistrand 2006, p. 83). Even prior to the democratisation of agile practices, studies have shown that avoiding dogmatic approaches to engineering methodologies can yield remarkable results at the end of the day for flexible organisations (Head 1994, p. 50). For software engineering teams aiming to deliver high quality products, the challenge of finding the best approach to their engineering processes is even more daunting because there doesn't seem to be an ultimate magic solution to satisfy all needs in any random setting (Ashrafi 2003, p. 679).

Consequently, previous research has suggested that firms need to actively look at fitting development processes, including novel agile approaches to their specific needs (Akbar *et al.* 2014, para. 1). It has for instance been shown that by having tailored processes specific to their environments, software engineering teams can reduce IT risks, stabilize projects lifecycles and ultimately facilitate the delivery of quality software products (*ibid.* 2014).

As previously stated, some empirical studies have highlighted practical reasons for the need to tailor agile practices to organisation specific needs. For example, in a 2006 European study conducted at Intel's Infrastructure Processor Division, it was observed that the organisation had opted to blend aspects of both Extreme Programming and Scrum as opposed to adopting a rigid implementation of either one of these agile approaches (Fitzgerald, Hartnett & Conboy 2006, p. 201). This was in part motivated by the fact that some XP practices were deemed superfluous to the organisational context if one considers for instance that (*ibid.* 2006):

- The planning game is better suited for Scrum settings.
- Continuous Integration is not simple to implement and can in certain cases require complex software suites from external vendors.
- There is an inherent difficulty in imposing working hours in multinationals which employ workers who collaborate in different time zones. This is especially problematic for large enterprises but can affect smaller organisations which outsource some of their engineering value chain offshore.
- It is also very difficult to implement some agile precepts such as the constant involvement of end users as in a large organisation such as Intel, it can appear

nonsensical to have customers on site when during early conceptual stages, hardly anything is formally defined in typical agile settings.

During the 2006 study mentioned above, a nuanced key finding for delivering quality end products was that the organisation didn't simply discard the practices staff didn't like, but rather that leadership carefully considered all practices and only eliminated the ones which were deemed impractical to implement in a restrictive manner (Fitzgerald, Hartnett & Conboy 2006, p. 206). What researchers seemingly observed to be an underlying success factor at Intel was that leadership opted to leverage Scrum and Extreme Programming practices for the organisational macro level, but then focused on customising these practices on a micro level for individual projects (*ibid.* 2006).

More recently, some case studies have attempted to find practical adaptable approaches to implementing agile methodologies and processes. Some of this research has even produced certain guidelines for the tailoring of agile processes as described in table 3-7 (Rolland, Mikkelsen, & Næss in Svorstøl 2017, p. 20) :

Table 3-7: Proposed guidelines for tailored agile as described by (Rolland, Mikkelsen, & Næss in Helen & Tracy 2016, p. 250)

Team Coordination Improvement	**Encouragement of novel practices**	**Adapt the project to disruptive changes**	**Adjust sprints content**
This guideline suggests organisations attempt to facilitate the establishment of domain specific communities of practices on one side and on the other to form shorter lived cross functional teams to fulfil organisational needs.	This guideline considers that in order to better customize agile precepts in large organisations, project members need to focus on tweaking and implemented practices which blend both technical aspects as well as the functional interdependencies which govern the lifecycle of the software products and artefacts being built.	It could also be suggested that projects teams cater for a ramping up period which enables system users to familiarize themselves with novel processes in their day to day interactions with a new software suite. Delivery	This guideline encourages teams to carefully consider the velocity at which information is delivered to system users as well as its pertinence. The objective is to not overwhelm the end users and allow for reasonable expectations in terms

		leadership needs in these cases to put an emphasis on quality training for customers.	of ramping up customers' technical know-how.

Through the discussion above, this chapter aimed to present approaches to customize agile engineering practices in modern software projects. In current times, such endeavours are better enhanced with the utilisation of related technology tool set to support the execution of agile projects. Subsequently, as the case study component conducted for this research is based at a small South African technology start-up, the next section of this thesis proceeds to introduce enabling technologies for agile settings by first contextualising the local scene.

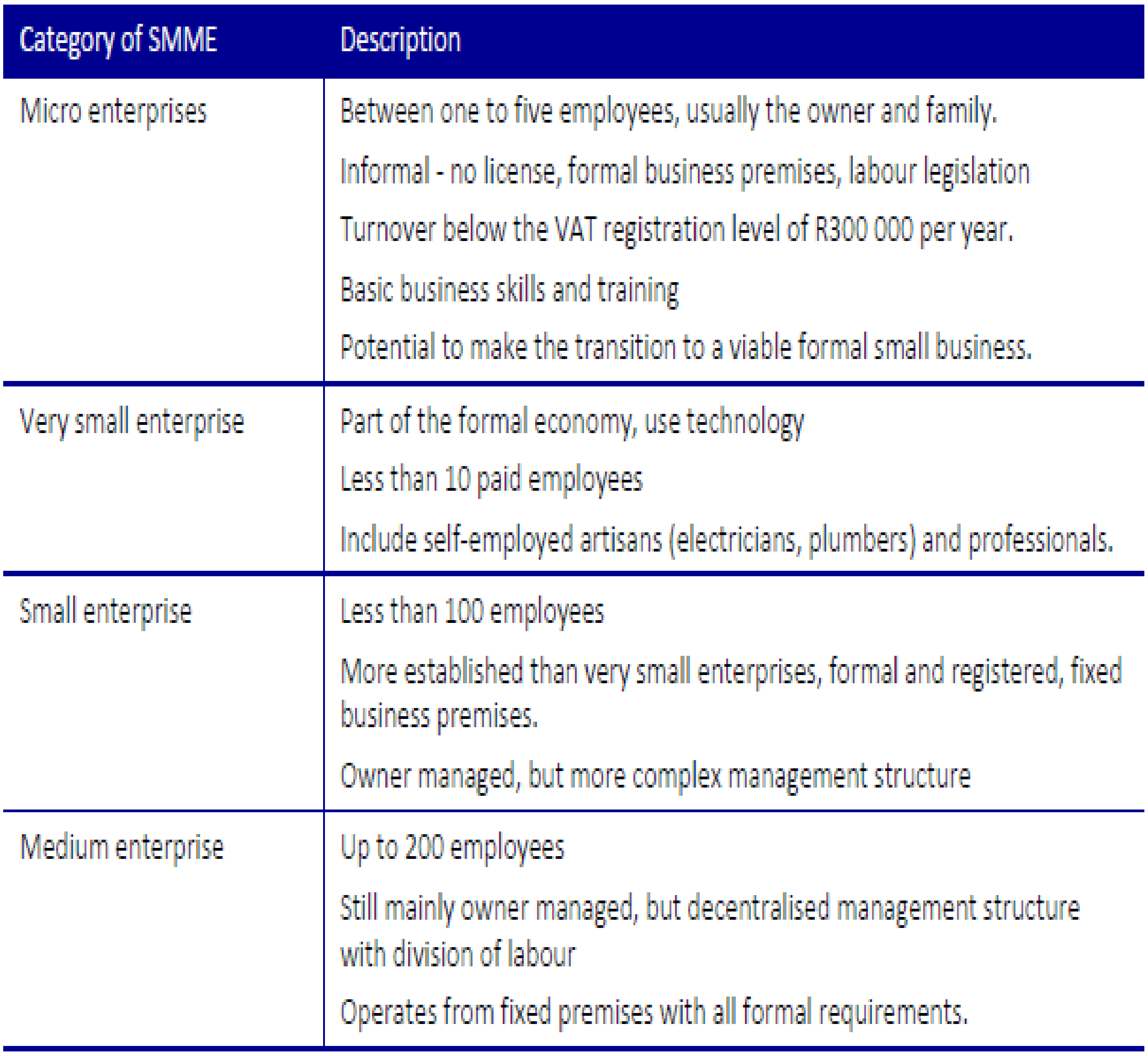
3.3 Disruptive agile technological realities

3.3.1 Understanding the role of technology in the South African SMME context

On the international stage, empirical evidence tends to indicate that small, micro , and medium sized enterprises (SMMEs) are key in the battle against unemployment (ITU 2016, p. 1). For example, the International Telecommunications Union estimates that these enterprises globally contribute up to 70% of GDP estimates (*ibid.* 2016). Furthermore, many African studies have also advanced a correlation between the positive performance of SMMEs and accelerated economic growth (Okpara & Wynn 2007, para. 2). In Nigeria for example, despite the high levels of poverty, small to medium sized entities provide more than seventy five percent (75%) employment (Kumalo & Poll 2015, p. 140). Other studies have even suggested that SMEs in Nigeria account for more than 80% of the local workforce (Owoseni & Twinomurinzi 2018, p. 1).

More contemporary local research has also shown that while SMMEs are dynamic players in most modern industrial nations, this is even more accurate emerging economies such as South Africa (Kumalo & Poll 2015, p. 140). As the importance of larger enterprises is vital in ensuring technology infrastructure and ecosystems are viable, SMMEs if afforded the opportunity to operate in enabling environments, can contribute to lead in terms of innovation and market transformations (DTPS 2018, p. 20). To date, academia consensus therefore seems to point towards SMMEs being agents of economic dynamism and growth.

In South Africa, the National Small Business Act distinguishes enterprises according to their sector of activity and further uses different thresholds to determine what is considered "small sized" (ILDP 2014, p. 12). Although these categorisations are formally described in the Act, the denomination for these categories are used inconsistently by state agencies, local sector databases and academia itself (*ibid.* 2014). Figure 3.5 describes these categorisations in the South African context:



Category of SMME	Description
Micro enterprises	Between one to five employees, usually the owner and family. Informal - no license, formal business premises, labour legislation Turnover below the VAT registration level of R300 000 per year. Basic business skills and training Potential to make the transition to a viable formal small business.
Very small enterprise	Part of the formal economy, use technology Less than 10 paid employees Include self-employed artisans (electricians, plumbers) and professionals.
Small enterprise	Less than 100 employees More established than very small enterprises, formal and registered, fixed business premises. Owner managed, but more complex management structure
Medium enterprise	Up to 200 employees Still mainly owner managed, but decentralised management structure with division of labour Operates from fixed premises with all formal requirements.

Figure 3.5: South Africa SMME categorisation

From a macroeconomic perspective, indicators have traditionally highlighted the importance of SMMEs on the local economy. Studies have for instance shown that a majority of formal South African business entities could be categorised in the small to medium enterprises category (Robert in Kumalo & Poll 2015, p. 140). Moreover, this kind of enterprises were estimated to contribute up to 57% to the local Gross Domestic product (*ibid.* 2015).

Recent reports describe an environment where SMMEs contribute more than forty percent (40%) to the national Gross Domestic Product (BER, 2016, sec. Executive Summary). Table 3-8 broadly summarizes key indicators of the South African SMME environment:

Table 3-8: SA SMMEs figures (BER 2016, pt. Executive Summary)

CHARACTERISTICS	**2015, 2ND QUARTER**
SMMEs identified total	2 251 821
Formal SMMEs total	667 433
Informal SMMEs total	1 497 860
Portion of SMMEs owners in the context of local employment figures	14%
Trade and accommodation SMMEs	43%
Community services SMMEs	14%
Construction sector SMMEs	13%
Financial and business services SMMES	12%
GDP contribution	42%
Previously disadvantaged communities' ownership portion	34%
Portion operated with less than R30000 annually	7%

In the South African context, technology is facilitating the ability for SMMEs to compete on the global scene (Oxford 2015, para. 1). It provides local organisations exposure to the global marketplace, lowers the barriers to entry while significantly cutting costs and thus allowing business leaders to evolve their offering from conceptual stages to effective market delivery (*ibid*. 2015). Ultimately, the adoption of ICTs by small and medium enterprises constitutes a promising dimension of ubiquitous growth (Gumbi & Mnkandla 2015, p. 2).

Oxford (2015) cites Internet Solution’s product manager Rindhir Singh who advances that: “*Small businesses are seeking enterprise class service regardless of their size”*. This state of affairs has led business owners to expect better customizable enterprise-level solutions to boost growth (Oxford 2015, para.2). One could link the assertion above with Gartner’s (2016) projected spending increase on enterprise grade software. In effect, this presents an opportunity for technology firms and organisations to propose innovative, tailored quality software solutions. This is especially true for SMMEs in general and the technology start-ups specialized in delivering software product and services to market. The next section hence proceeds to a discussion on the local software industry in terms of the start-up environment.

3.3.2 Micro-economic perspective

A survey on South Africa’s local start-up ecosystem provided the following key insights (Ventureburn 2015):

- Approximatively eighty-eight percent (88%) of start-ups have five employees or fewer. This fits with the current research’s scope of study which focuses on start-ups which fit the SMME categorisation
- Almost thirty-eight percent (38%) of start-ups have no employees and comprise of just the founder.
- Most tech start-ups operate in the computer and software services niche.
- Most founders of these organisations tend to be software engineers or have some sort of technical background.
- About forty-nine percent (49%) percent of start-ups are run from homes and garages, twenty-three percent (23%) are run from rented offices. Most companies operate in this manner because of an underlying need to bootstrap as well as cut costs.

- Only seventeen percent (17%) of start-ups surveyed indicated that they were profitable.

These key identifiers of South African start-ups are not showstoppers to their growth. In practice, these businesses can be competitive locally and abroad.

Globally, the second quarter of 2016 for instance posted year-over-year and quarter-over-quarter growth of 4.2% within the various technology sub-sectors (PWC 2016b, sec. Executive Summary). In the third quarter of 2016, the projections included (PWC 2016c, sec. Executive Summary):

- More than Twenty-Four per cent (24.2%) growth for the Internet Market.
- Increased growth for Software products (2.5%).
- Increased growth for Software services (7.5%).

These international growth figures present South African tech start-ups specializing in software delivery with a huge opportunity to service the global and sub-Saharan markets. Additionally, the local business demographics of start-ups seem to be mostly occupied by firms operating in the computer and software services sector (Ventureburn 2015).

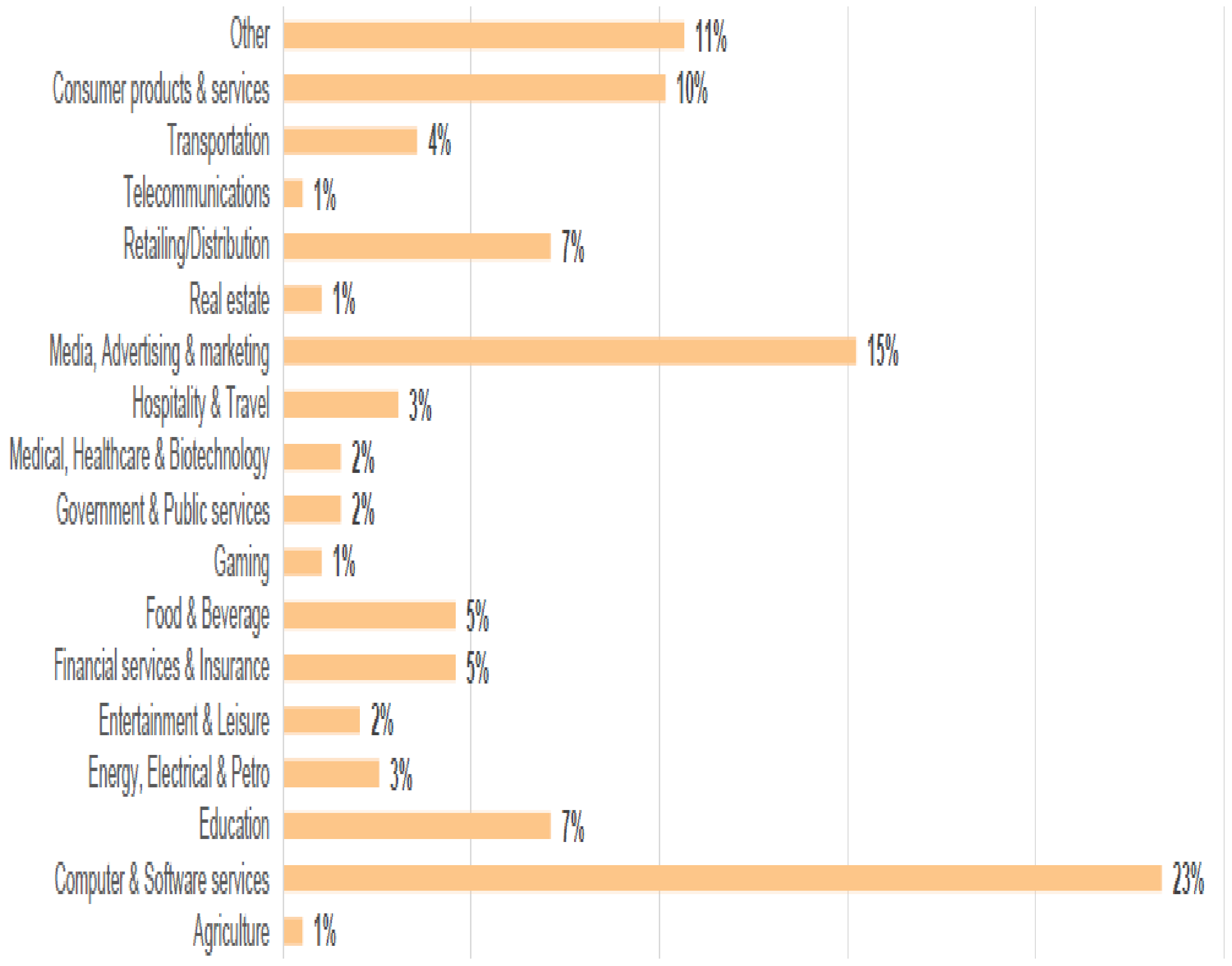


Figure 3.6: Technology start-up activity sectors (Ventureburn 2015)

Although anterior to PWC's projections at the end of 2016, figure 3.6 seems to show that the lion's share of the local start-up environment is dominated by organisations which operate in the software services sector. Nonetheless, SMMEs in general and start-ups specifically face two key challenges in South Africa (Ventureburn 2015). These include:

1. *A lack of funding opportunities:* Endeavours for the financing of Small Medium and Micro Enterprises (SMMEs) are often timid. This is because failure rates experienced amongst start-ups in South Africa is alarming. In effect, around 56% of start-ups are self-funded from the start (Ventureburn 2015). Not having access to venture capital funding opportunities is an obvious obstacle to innovation on the local software market, and hence hampers the industry potential for greatness in many instances.
2. *A lack of revenue, thus lack of marketing:* Most start-ups are bootstrapping and cannot afford traditional forms of marketing (Ventureburn 2015).

Furthermore, although financial constraints tend to limit the adoption of modern expensive technology implementations, cloud technology has given smaller firms access to technology with enterprise level capability without the traditional complexity and incurred costs (Oxford 2015, para. 3). In this light, Chapter Three now proceeds to a conversation on the technological means local firms have at their disposal to conduct their activities in a cheaper, leaner and disruptive manner.

3.3.3 The emergence of cloud computing

Cloud services can be described as the delivery of IT resources such as computing nodes, databases, proprietary software access, infrastructure etc. through the internet and often charged through a pre-paid model.

In modern times, the Cloud space has evolved from its experimental beginnings to provide enterprises with a platform for real business value. In effect, while organisations worldwide have understood the importance of heavy investments in Digital transformation endeavours, Cloud based services have emerged as a paramount foundation to achieve this expected evolution (EMA 2018, p. 1).

Cloud technology presents a paradigm shift for large, small and medium sized enterprises alike as it facilitates the operations and rendering of services for organisations tackling dealing with IT assets and increased resourcing costs (Hinde & Van Belle in Gumbi & Mnkandla 2015, p. 1). This new computing alternative has improved the way in which application are being developed by engineers around the world. Furthermore, it has enabled organisations to streamline their development lifecycles, improved productivity in engineering teams while fostering and nurturing collaboration between stakeholders (Mckenzie, Tee & Denman 2013, p. 2).

In terms of the South African industry more specifically, cloud computing is often understood as a paradigm which conveniently facilitates the on demand provisioning of computing and networking assets without requiring ostensible participation from service providers (Dillon et al. in Gumbi & Mnkandla 2015, p. 5).

The maturity of enterprise grade cloud technology has resulted in the democratization of cloud services procurement. In practice, purchasing services from cloud providers instead of buying cumbersome hardware and software stacks to self-manage is an attractive proposition for most businesses. This has led to IT leaders increasingly leaning towards the adoption of cloud strategies as they enable (Fox, Shields & Pang 2012, p. 3):

- The acceleration of time to market for organisations aiming to keep ahead of competitors.
- The ability to use a cheaper Pay as you go model.
- Replacement of capital investments by operating expenses.
- Scalability of resources according to usage requirements.
- The ability to free up operation resources and staff by utilizing vendor support.

From a senior management perspective, the cloud model is also understood to be mature enough to present real opportunities for organisation who embrace it. Thus, businesses which adopt an explicit cloud strategy integrate cloud services and facilitate business stakeholders involvement in a structured manner (Villatore-Silva 2012, p.3).

From a local start-up standpoint, research has shown that ICT in general and the cloud in particular gives early adopters the edge regardless of their location in emerging or industrial economies (Oxford 2015, para. 5). As such, software developers have a large array of options in terms of software architectures they can implement which will be supported on innovative, flexible and cutting edge cloud platforms (Homann & Mckenty 2018, p. 3).

This leaves local software engineering organisations with the realization that the ball is in their court, and that with the emergence of cloud technologies, the sky is the limit when it comes to disrupting local or international markets.

Modern agile software engineering teams have for many embraced the opportunity cloud technology provides them. They nonetheless typically require five pillars to deliver innovative solutions to market:

- A reliable source code control system.
- Robust infrastructure.
- An agile project management framework.
- A continuous integration system for smooth DevOps.
- A communication system which facilitates meaningful and insightful collaboration throughout teams.

The next sections proceed to discuss modern disruptive incarnations of these cornerstones for agile teams.

3.3.4 Source Control in the Cloud: *the case of Github.com*

Software systems are built from source code which evolves throughout the development lifecycle. Changes to the source files containing the code software is built from have historically presented challenges to engineering teams. Previous research has enumerated problems software teams had to consider in this regard (Rochkind 1975, p. 364):

1. The amount of storage required to store source files tends to increase depending on the version of the code teams are working on. The space requirements can exceed what was initially identified during infrastructure planning stages. Requirements that were hence not initially considered have the potential of rendering infrastructure obsolete once additional features are taken into account.
2. Fixes made to a specific version of a module are in certain cases not properly propagated to other versions. This can lead to inconsistencies during testing or the final product delivery.
3. When changes occur, it can be challenging to determine exactly what changed when and by which engineer.
4. When users encounter production problems, it is difficult to figure which version of the software they are using when no proper versioning system is in place.

Ultimately, software engineering teams had to objectively assess their needs in terms of archiving old versions of files and directories while having the capacity to examine logs about changes to source files over time (Collins-Sussman, Fitzpatrick & Pilato 2008, p. 13). This led to the popularity of version control tools such as subversion as they facilitated collaboration within software engineering teams. In practice, such tools enabled engineers to collaborate with colleagues on documents over a network and keep track of who made changes to code and other documents during a project (*ibid.* 2008). Version Control Systems (VCS) have naturally improved and evolved over time. The most prominent and contemporary incarnation of these systems is GIT. The next section proceeds to an overview of this system.

GIT is a user favourite source control system as it gives users flexibility and strongly supports agile development practices (Just *et al*. 2016, p. 401). It mainly differs from other version control platforms by the way it conceptualises source code data. While previous approaches to source control have traditionally focused on source code as a set of files updated overtime, GIT based system considers data as minimized snapshots of a reduced file system (Chacon & Straub 2014, p. 13).

GIT provides engineering teams the opportunity to increase agility because it provides (*ibid.* 2016):

- ***An intuitive and simple branching mechanism.*** GIT systems allow users to easily create remote and local branches to ensure work isolation and the ability to merge at a later stage. This facilitates the parallelization of tasks and the ability of team members to focus on their assigned features.
- ***The ability to easily commit code and revert locally***. GIT allows engineers the facility of committing more often and creating private local restore points. This can be deemed as one of the most powerful advantages the system affords programmers.
- ***Merge operations at fine level of granularity***. Developers are afforded the freedom of accepting and rejecting individual changes as opposed to batches which allows speedy detection of defects and hence facilitates revert operations.

All these features of the GIT system provide organisations an immensely powerful source control tool. This is made even more attractive to engineering teams if one considers the quasi free open source cloud offerings which leverage this system. In this light, the next section discusses such an offering in the form of GITHUB.com.

GitHub is a software code-hosting web service which provides a web interface for the management of GIT systems. This service is considered to be the primary online work platform for developers to host their own software source code and contribute to other open source projects (Wu *et al*. 2014, p. 266). It also facilitates collaboration and foster agile practices in software projects by offering social and project management capabilities to software teams (Longo & Kelley 2015, p. 1). GitHub is basically a tool which enables every engineer the possibility to either contribute on open source projects or within their organisations (Todorovic 2015, para. 3) . It is a highly respected collaboration platform which additionally is used by big corporates such as IBM, Netflix, Facebook, Google etc.

This platform provides local engineering teams with a huge boost in the sense where they can afford to have a potentially infinite number of projects hosted in the cloud for them at no charge. They can also leverage the social features of the platform to interact with other engineers who can assist in collaborating on innovating solutions where interest is mutual. For organisations which deem certain projects too sensitive, code can be hosted in private repositories at a cost but without losing any advantages of typical GIT based system.

After source code is built into a runnable artefact, it is still dependant in certain cases not only on end user devices, but also on an underlying infrastructure stack to run. Agile teams rely on such infrastructure to host the backend services client interfaces interact with for services to be rendered. In this light, the next section discusses the cloud infrastructure paradigm with an introduction to Amazon's cloud services.

3.3.5 Infrastructure as a service in the cloud: *the case of Amazon Web Services*

With Infrastructure as a Service models, organisations are typically able to purchase on a per usage basis for services which range from automation control, storage, networking facilities etc. which are all owned and managed by a service provider who takes ownership for the underlying hardware hosting customer applications (Wahaballa *et al.* 2015, p. 214). Such models usually provide customers fully customizable managed infrastructure to fulfil their software execution needs (Lampe 2011, p. 69). For businesses interested in harvesting this powerful technology, Amazon Web Services (AWS) by Amazon is such an offering.

Amazon proposes managed infrastructure services as part of its AWS cloud computing offering. One of the key benefits of this service is that it offers large enterprises and start-ups alike the opportunity to replace up-front capital infrastructure expenses with low variables that scale in conjunction with business growth and increasing requirements (Varia & Mathew 2017, p. 1). Amazon supports a large array of cloud offerings which include Infrastructure, Platform and Software as a Service. The provider also offers the ability for businesses to choose between the hybrid, full cloud, or on-premise deployments strategies. Another benefit of using the Amazon platform is that if offers a free tier for 12 months where businesses can leverage AWS selected services at no charge. This is attractive for small businesses with low compute requirements who just need to get their products of the ground or simply require lab environments to test their new technologies before delivery to markets.

3.3.6 Managing agile projects in the cloud: *the case of Atlassian's JIRA*

Agile software engineering teams perform development in an iterative and incremental manner. This methodology is used by modern organisations because user requirements are not static, and teams need to adapt to this reality. To effectively take advantage of the most recent approaches of running software projects, project managers are increasingly required to be comfortable with Agile Project Management. This way of managing modern software projects provides project managers with methodologies and tools which will help them tackle difficulties related to mass evolutions in terms of technologies and ever changing customer needs (Othman *et al.* 2010, p. 102).

In terms of planning tools in agile oriented settings, Atlassian is an Australian cloud tooling service provider which is today considered a world leader in its tool chain offering including widely adopted JIRA agile project managing service. Figure 3.7 describes Gartner's magic quadrant for enterprise agile planning tools:

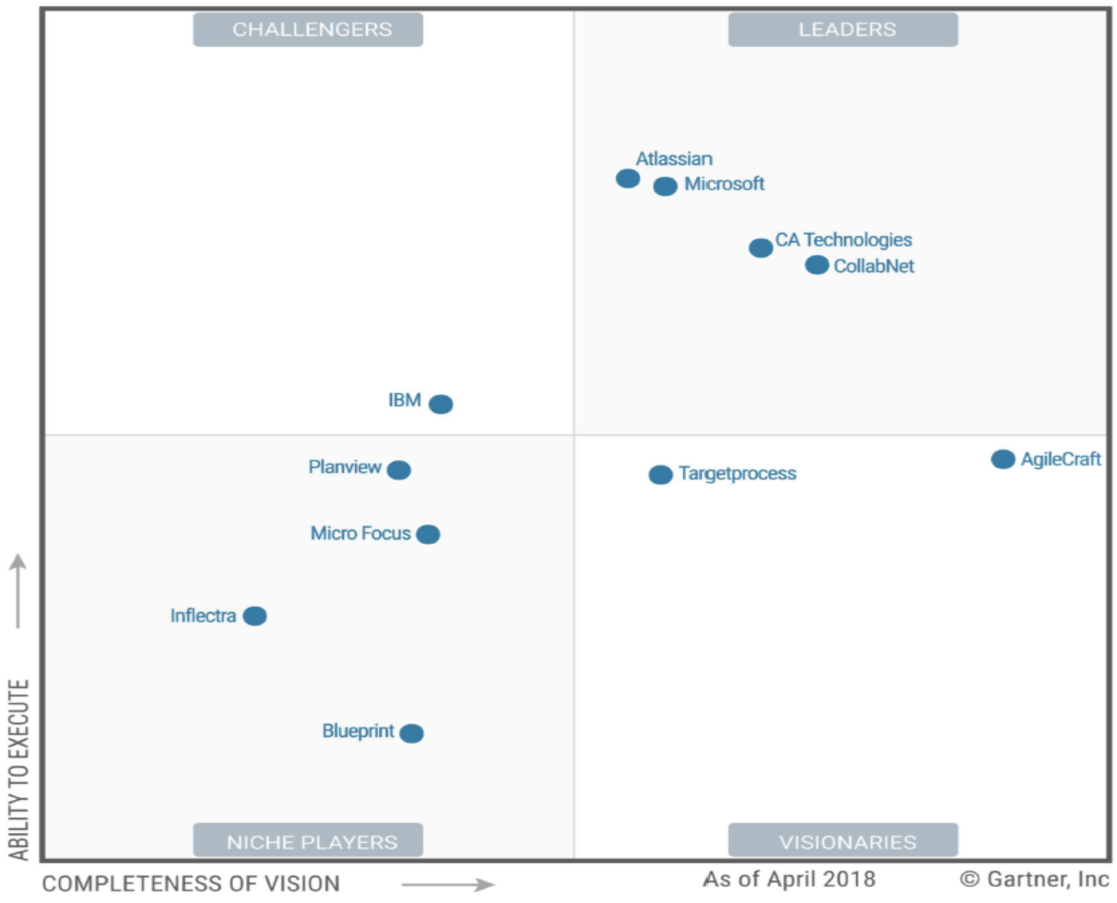
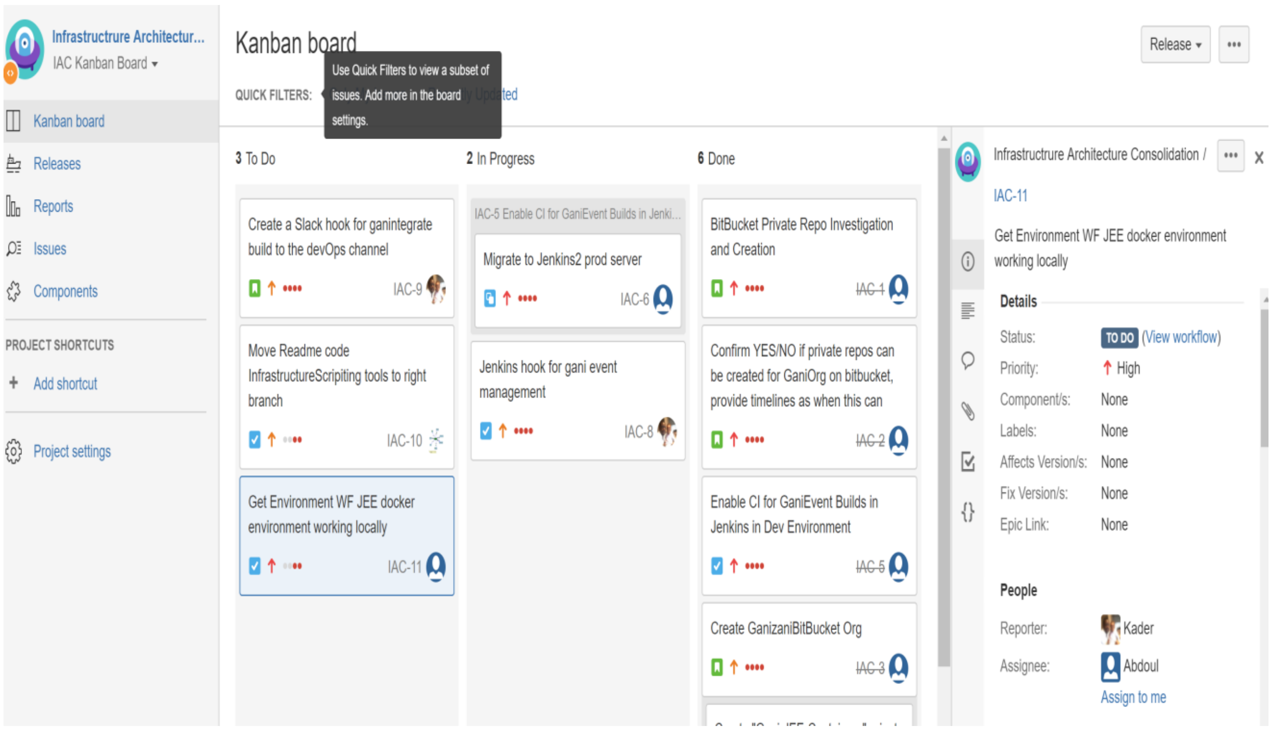


Figure 3.7: Magic Quadrant for Enterprise Agile Planning Tools (Mann, Murphy & West 2018, fig. 1)

As part of Atlassian' agile and DevOps related offering which includes software such as Bitbucket, Bamboo, Confluence and HipChat etc. JIRA is an open source collaboration platform for agile teams which facilitates agile implementations in organisations.

The system offers a cloud web interface where users can capture project issues and track them. Throughout the evolution of the platform, new features have been provided where scrum masters and engineers are able to capture user stories, track tasks progress on a Kanban board etc. JIRA also offers a task board view, a planning board and many more (Sarkan, Ahmad & Bakar 2011, p. 408). Figure 3.8 shows a Kanban board screenshot one would find on the interface:



Use Quick Filters to view a subset of issues. Add more in the board settings.

3 To Do

Create a Slack hook for ganintegrate build to the devOps channel IAC-9

Move Readme code InfrastructureScripting tools to right branch IAC-10

Get Environment WF JEE docker environment working locally IAC-11

2 In Progress

IAC-5 Enable CI for GaniEvent Builds in Jenki...

Migrate to Jenkins2 prod server IAC-6

Jenkins hook for gani event management IAC-8

6 Done

BitBucket Private Repo Investigation and Creation ~~IAC-1~~

Confirm YES/NO if private repos can be created for GaniOrg on bitbucket, provide timelines as when this can ~~IAC-2~~

Enable CI for GaniEvent Builds in Jenkins in Dev Environment ~~IAC-5~~

Create GanizaniBitBucket Org ~~IAC-3~~

Infrastructrure Architecture Consolidation / IAC-11

Get Environment WF JEE docker environment working locally

Details

Status:	TO DO (View workflow)
Priority:	High
Component/s:	None
Labels:	None
Affects Version/s:	None
Fix Version/s:	None
Epic Link:	None

People

Reporter:	Kader
Assignee:	Abdoul
	Assign to me

Figure 3.8 JIRA Kanban board view

To simplify workflow for agile teams which prefer traditional scrum approaches to managing their projects, JIRA encourages collaboration between stakeholders during sprint planning and execution:

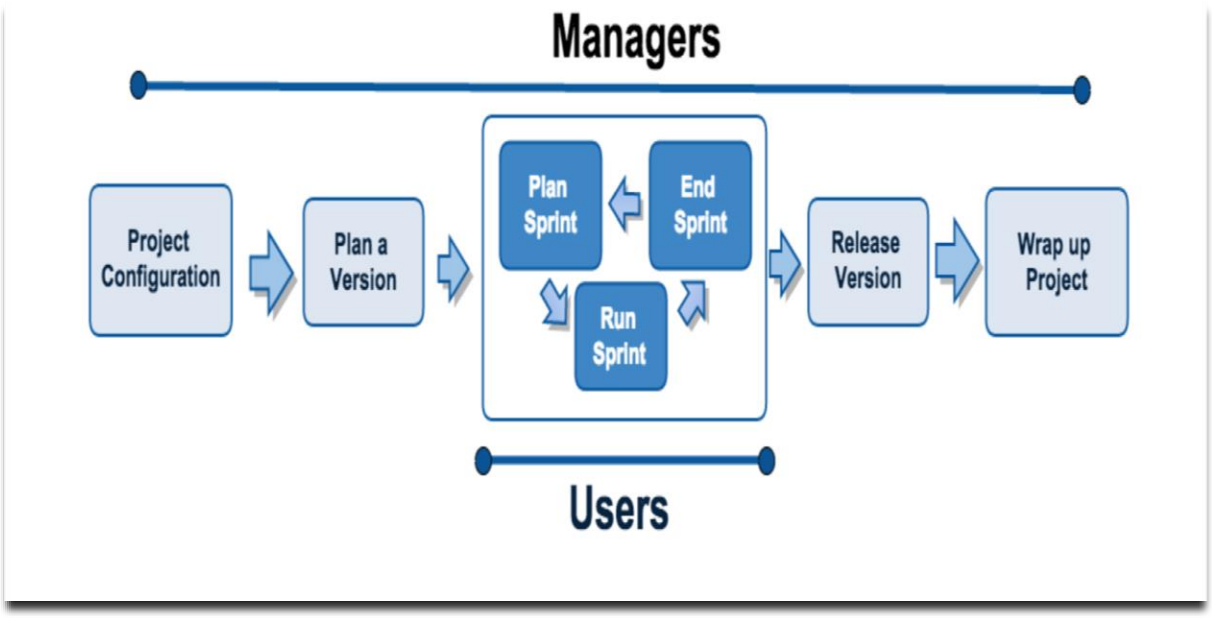


Figure 3.9 Simplified JIRA Scrums (Atlassian 2015, p. 5)

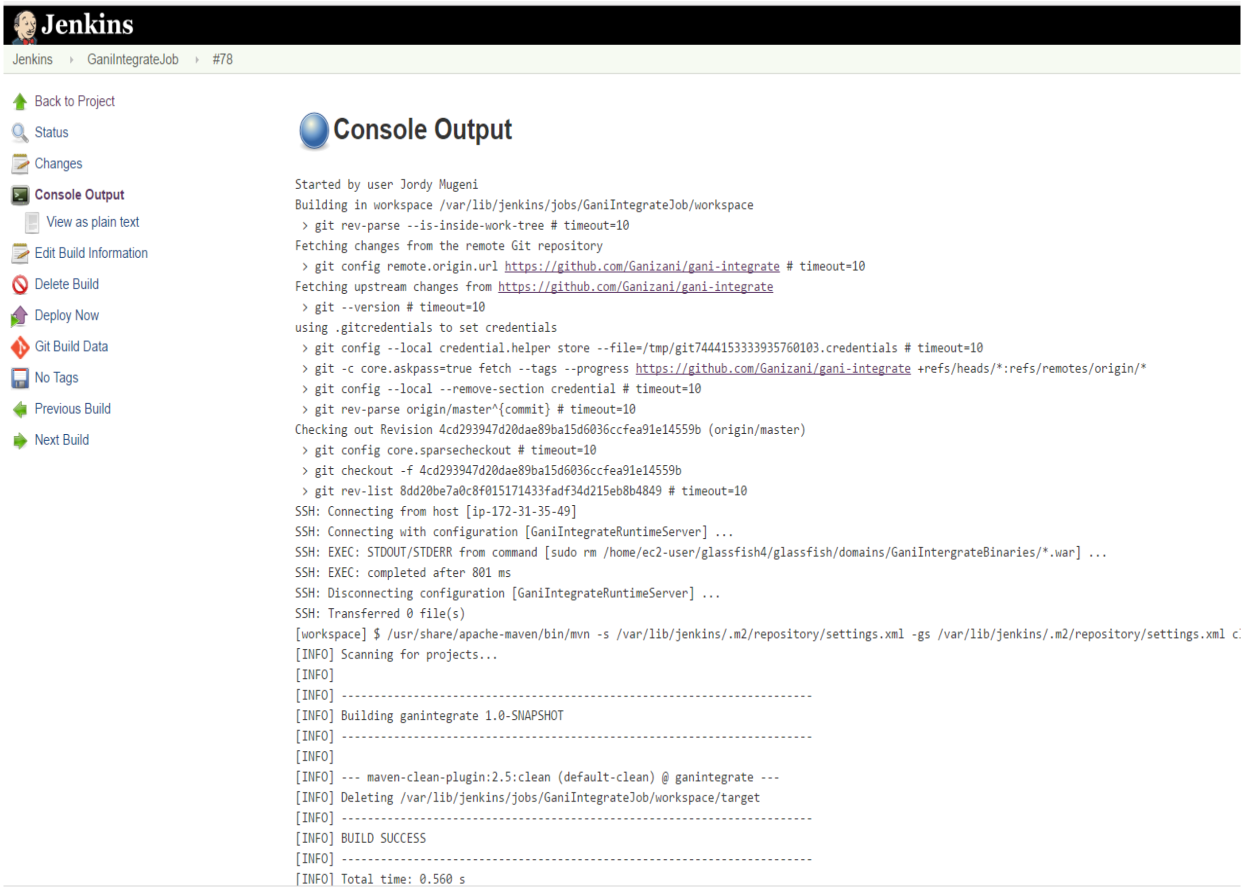
JIRA additionally offers integration to other agile facilitating tool such as GITHUB mentioned above. Administrators can for example link tickets on a board to source code repository issues and can hence track physical code progress and updates to the version control system.

3.3.7 Agile continuous integration: *the case of Jenkins*

The Time to Market of an innovative software product has an immense impact on the likelihood of its survival in current markets. This is most true when one considers new technologies which need to be delivered to consumers with shorter than average attention timespans. In the current knowledge economy, what really differentiates the product is not only quality in itself, but also the speed at which it can evolve (Armenise 2015, para. 1).

Continuous Integration (CI) can be used as a complement to Test Driven Development. It typically involves engineers successfully performing unit tests execution locally before committing code to upstream repositories. Furthermore, in CI environments, a server is used to not only execute unit tests, but also build and ship source code artefacts to intended targets.

Jenkins is an Open Source CI platform which facilitates the automation of code builds as well as unit and integration testing. The application is an open source java platform which provides customization and extensibility by leveraging a plugin extension architecture (Armenise 2015, p. 24). Figure 3.10 illustrates the Jenkins user interface as well as console verbose from a successful build:



```
Started by user Jordy Mugeni
Building in workspace /var/lib/jenkins/jobs/GaniIntegrateJob/workspace
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/Ganizani/gani-integrate # timeout=10
Fetching upstream changes from https://github.com/Ganizani/gani-integrate
 > git --version # timeout=10
using .gitcredentials to set credentials
 > git config --local credential.helper store --file=/tmp/git7444153333935760103.credentials # timeout=10
 > git -c core.askpass=true fetch --tags --progress https://github.com/Ganizani/gani-integrate +refs/heads/*:refs/remotes/origin/*
 > git config --local --remove-section credential # timeout=10
 > git rev-parse origin/master^{commit} # timeout=10
Checking out Revision 4cd293947d20dae89ba15d6036ccfea91e14559b (origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 4cd293947d20dae89ba15d6036ccfea91e14559b
 > git rev-list 8dd20be7a0c8f015171433fadf34d215eb8b4849 # timeout=10
SSH: Connecting from host [ip-172-31-35-49]
SSH: Connecting with configuration [GaniIntegrateRuntimeServer] ...
SSH: EXEC: STDOUT/STDERR from command [sudo rm /home/ec2-user/glassfish4/glassfish/domains/GaniIntergrateBinaries/*.war] ...
SSH: EXEC: completed after 801 ms
SSH: Disconnecting configuration [GaniIntegrateRuntimeServer] ...
SSH: Transferred 0 file(s)
[workspace] $ /usr/share/apache-maven/bin/mvn -s /var/lib/jenkins/.m2/repository/settings.xml -gs /var/lib/jenkins/.m2/repository/settings.xml c
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building ganintegrate 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ ganintegrate ---
[INFO] Deleting /var/lib/jenkins/jobs/GaniIntegrateJob/workspace/target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.560 s
```

Figure 3.10: Jenkins build verbose

This tool is attractive to agile teams because it can be run with a simple java Web Archive file (WAR) or used on the various cloud platform offerings on market. Its intuitive easy to use interface is another plus. Ultimately, it facilitates building CI pipelines at no charge and is hence a game change for start-ups with limited financial resources.

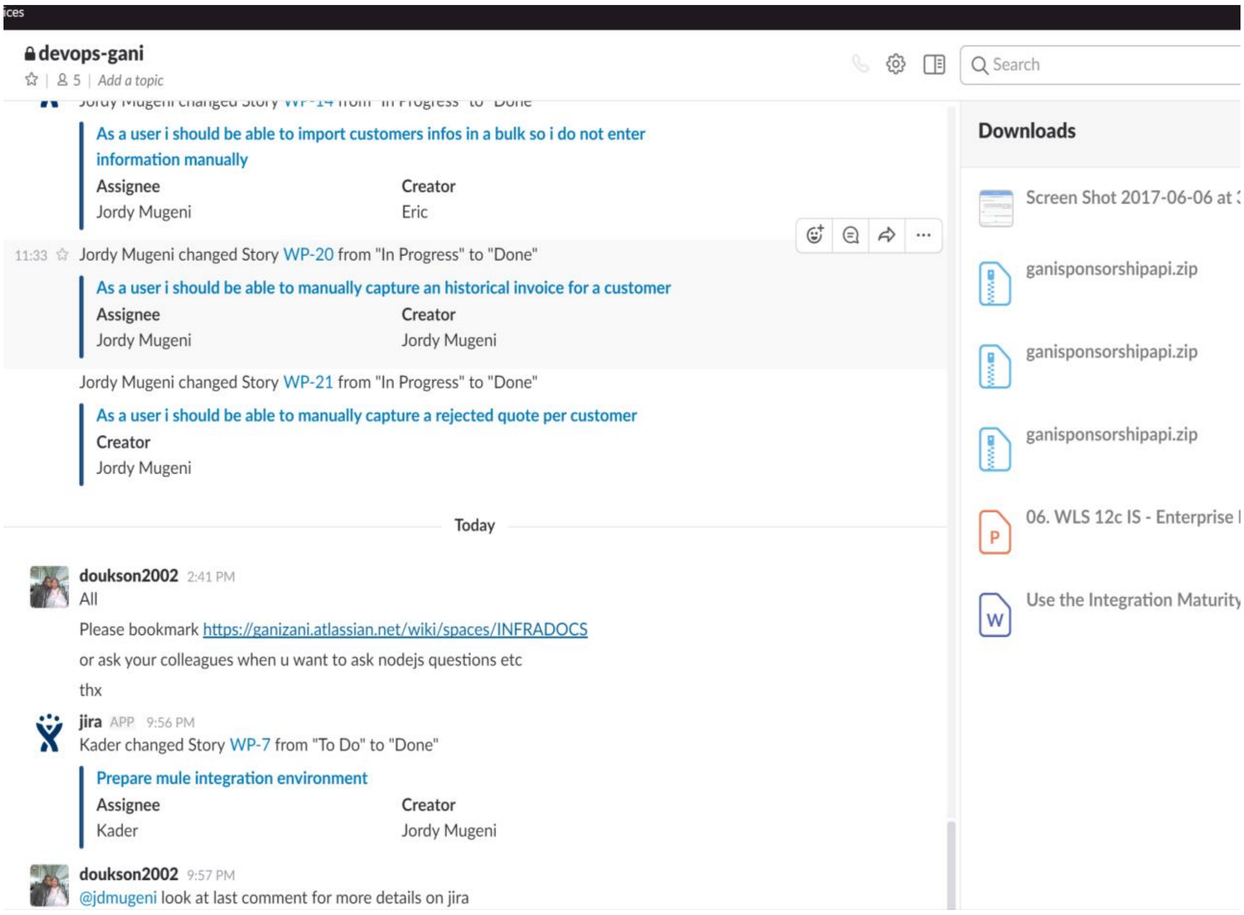
3.3.8 Agile enabling collaboration tools: *ChatOps and the case of Slack*

ChatOps facilitates communications between project team members by integrating their tools, following established processes while enabling automated workflows (Regan 2016, para. 1). Agile teams have always been conversation driven, but ChatOps is best described as its digital-age manifestation. The most appealing aspect of this new paradigm is its chat aspect. It represents a new way to capture the collective knowledge of a team. This has resulted in a revolution in the way agile teams collaborate and deliver innovative products to marketplaces worldwide (Regan 2016, p. 1). In the next paragraph, a revolutionary ChatOps product is introduced: Slack.

Slack is a tool which offers users a digital workplace for communication and collaboration in an intuitive manner rarely seen. It provides a desktop client for most operating systems or can simply be used in standard JavaScript enabled web-browser. Its most basic users are:

- A Team Owner.
- Administrators.
- Team Members.
- Guests.

The way the system basically works is that a team owner creates a team, recruits' administrators for the management of these teams and performs the onboarding functions of members. One of the reasons this tool is so attractive to agile software engineering teams is that it de-clutters information and mimics a publish-subscribe model where projects or topics of interest are organized into "Channels". A user is hence free to see only topics he or she is interested in and has been subscribed to. Another key attraction of slack is that it can be integrated to a team's JIRA system or even its Continuous Integration platform. The entire team or selected members can hence monitor new builds, deployments, tickets issued in real-time. Figure 3.11 illustrates such a scenario in slack:


devops-gani

Jordy Mugeni changed Story WP-14 from "In Progress" to "Done"

As a user i should be able to import customers infos in a bulk so i do not enter information manually

Assignee Jordy Mugeni | Creator Eric

11:33 Jordy Mugeni changed Story WP-20 from "In Progress" to "Done"

As a user i should be able to manually capture an historical invoice for a customer

Assignee Jordy Mugeni | Creator Jordy Mugeni

Jordy Mugeni changed Story WP-21 from "In Progress" to "Done"

As a user i should be able to manually capture a rejected quote per customer

Creator Jordy Mugeni

Today

doukson2002 2:41 PM
All
Please bookmark https://ganizani.atlassian.net/wiki/spaces/INFRADOCS
or ask your colleagues when u want to ask nodejs questions etc
thx

jira APP 9:56 PM
Kader changed Story WP-7 from "To Do" to "Done"

Prepare mule integration environment

Assignee Kader | Creator Jordy Mugeni

doukson2002 9:57 PM
@jdmugeni look at last comment for more details on jira

Downloads

Screen Shot 2017-06-06 at
ganisponsorshipapi.zip
ganisponsorshipapi.zip
ganisponsorshipapi.zip
06. WLS 12c IS - Enterprise
Use the Integration Maturity

Figure 3.11: Slack interface reporting on JIRA ticket updates

3.4 Limitations to siloed usage of agile tools

As discussed earlier, software engineering teams have a multitude of options when it comes to technology which facilitates the implementation an agile philosophy throughout their organisations. In effect, recent studies have shown that organisations require state of the art enabling technologies to improve their decisions making processes and effectively run their engineering projects (Pospieszny 2017, p. 5). From the Source control systems, through infrastructure, project management and CI tools; the cloud offers endless opportunities to teams regardless of their financial means.

South African start-ups can leverage these technologies, but an issue remains. Despite the ability to use all these tools and the reports as well as comprehensive data they provide; the sheer amount of this information can be overwhelming. This can lead to information generated being used in silos. In effect situations arise where the DevOps team could be tempted to only look at CI reports, core development staff members only at GIT related data, scrum masters only at JIRA tickets etc.

Few dynamic start-ups with only a couple of employees could argue that they embrace this challenge or that such conundrums do not apply to them because they use only a small amount of these tools. Others could argue that they use only tools such as JIRA to have a single view of information. A counter argument to this point is that this would in fact defeat the purpose of having unique functions available in specific products. The fact remains that source control, infrastructure, DevOps and ChatOps are aspects of the knowledge economy software organisations must deal with if they are to survive in these competitive times. Having identified gaps and limitations in the silo usage of this technology, the current research therefore suggests integrating the technological tools discussed during earlier sections into a comprehensive framework in order to achieve:

- The effective integration of agile technologies into a consolidated web portal to aggregate disparate sources of information in the running of software projects.
- The leveraging of this consolidated information for improving agile (engineering and project management) processes in order to facilitate the delivery of quality software.

3.5 Chapter Three summary

4. CHAPTER FOUR: RESEARCH METHODOLOGY

At the term of reviewing literature, researchers typically aim to establish an underpinning theoretical framework (Sekaran 2003, p. 86). This entails conceptualizing a model inspired by previous studies and informed by the researcher's own perspective which highlights connections between key concepts relevant to the research problem at end (*ibid.* 2013). The aim of this exercise is to describe the scientific approaches used to gather, aggregate, process and interpret data during research.

After a discussion on epistemology, the current chapter proceeds to describe this study's research methodology and various paradigms available to academics in the Information Systems field. Subsequent sections of this thesis discuss how and why a practical case study was used to address the research objectives. Issues related to data collections techniques and research ethics are also addressed in throughout Chapter Four.

4.1 Background to epistemology

In the following section, background theory to this study is presented. The thesis then follows up with a discussion on foundational concepts of academic enquiry. This includes aspects of epistemology as well as the underpinning research paradigms.

4.1.1 Epistemological considerations

Epistemology typically refers to the theory of knowledge and how it has been acquired and sustained the test of time. This theory of knowledge is used to conceptually detail the contents of knowledge, its identified features and how it is applied in various contexts (Rescher 2003, p. 13). Epistemology also illustrates how a researcher approaches his quest for knowledge irrespective of the study's geographical location. In academia, the epistemological orientation of research has been observed to have a central influence on the inquiry itself (Mwansa 2015, p. 47).

When conducting research in sub-Saharan Africa, one cannot disregard the current globalisation context. Local researchers are often tempted by the difficult challenge of looking at the issue of epistemology through an exclusively African lens (Higgs 2010, p. 2114). A desire to mitigate the perceived latent effects of colonialism probably played a part in this situation, yet, one cannot ignore the impact of other cultural aspects such as the transmission of knowledge through oral tradition for example (*ibid.* 2010).

In this thesis, the researcher opted to adopt a traditional epistemological perspective as the dissemination of knowledge has been found amongst other factors to transcend culture and civilizations (Hirschheim 1992, p. 14). This is partly why the current chapter limits its discussion to traditional doctrines including positivism and interpretivism which were considered to fit best the current research objectives for reasons to be enunciated further along Chapter Four. In this vein, the next section proceeds to introduce positivism and interpretivism.

4.1.2 Discussion on positivism

The research context of western human sciences has largely influenced traditional Information Systems academia. When attempting to look at epistemology from a historical perspective. four central ages could be discussed (Hirschheim 1992, p. 14). Table 4-1 goes on to describe a historical synthesis of these phases.

Table 4-1 Condensed history of western epistemology (Hirschheim, 1992)

1620-1888: The hegemony of positivism	*This period where the bulk of intellectual discussion revolved around theology is viewed in a negative light by many observers. The philosophy writings of figures such as Aristotle and Plato were scarcely shared as most scientific questions where interpreted by clerics as fostering doubts in the existence of God.*
1844-1937: The emergence of anti-positivism	*During this period, men of science were alarmed by perceived shortcomings of positivist approaches to epistemology. They highlighted the fact that to give weight to questions of knowledge, one could not afford to ignore basics of human phenomena such as meaningful experience. Authors such as Kant emphasised the fact that when human values were given too much prominence in academic works, the results of such enquiries could not be deemed objective and contained in many cases inaccuracies which left some research devoid of any scientific credibility.*
1919-1937: The return of positivism	*This period yielded some epistemology advancements in what is today deemed as logical positivism. This approach which has greatly influenced contemporary work, aimed to give rational explanations to human reality. It informs the core of current hypothetic-deductive styled research.*

1926-1980: The opposition of contemporary critics	*Logical positivism was criticised by 20th century intellectuals who advanced that disregarding the history science could would yield more accurate knowledge. In practice, it has been observed that most scientific observations are in some shape or form informed by theoretical assumptions. In this sense, the assertions by logical positivism that true science entails the dominion of deductive reasoning and complete removal of human inductions was decried.*

In recent times, many researchers have mostly adopted positivist angles when attempting to validate formulated theories. The assumptions of studies conducted in such a manner is that subjects being observed and the researcher should perform their respective activities independently (Gaaloul & Molnar 2014, p. 61).

The current research assumes a partly positivist view during practical experimentation where an information system is designed and implemented for usage at a South African start-up. For this experimental part of the study, it is suggested that to reach the objectives identified in Chapter One, a technological tool could be implemented to complement existing systems used by project teams to facilitate the management of agile related activities during projects, and thus contribute to the delivery of quality software. The case study at GZ Consulting Services will be used to validate or not the assumption formulated above. Traditionally, a case study is well suited for positivist frameworks if one accepts that scientific enquiry should be based on what can be observed through the quest for empirical evidence (Gray 2013, p. 21).

For research exercises like the current one, previous studies have shown that qualitative approaches are also very efficient. This, as one needs to ensure that results obtained during experimentations such as the ones typically conducted during a case study are informed by sufficient literature (Gaaloul & Molnar 2014, p. 61) . In effect, for situations where researchers observe subjects, it is important to realise the impossibility of claiming any sort of absolute objectivity. In such scenarios, the researcher should not ignore the interpretative dimension of scientific enquiry.

As such, because qualitative methods are usually closely linked to interpretivism, in the next section, the current chapter proceeds to discuss issues surrounding the interpretivist doctrine.

4.1.3 Discussion on interpretivism

Interpretivist researchers propose that reality is a human social construct, and that as such, it should not be taken at face value. This philosophy considers that it is not realistic to expect absolute objectivity during the quest for knowledge. Interpretivism posits this notion because when an investigator interacts with subjects of his or hers study, all actors are inter-influenced, and this can drastically change preconceived notions, perceptions, attitudes or even the manner in which mundane activities are conducted (Walsham 1995, p. 376). Interpretivist point of views advocate for more nuance in analysing phenomena. For instance, many interpretivist academics believe that there is no direct bridge between observable subjects and the world outside their confines (Gray 2013, p. 22). As such, a researcher cannot use simple deductive reasoning to assign motives to subjects or causes to phenomena if he or she does not use an interpretative lens to understand the subtleties of knowledge gathered.

Mixed method research involve integrating multiple techniques by blending instruments such as surveys and experiments with traditional interviewing and focus group interactions (FoodRisC, 2016, para. 1). The present dissertation adopted these methods by conducting an experimental case study with development of an IS prototype, as well as collecting data by using an electronic survey, direct observations and semi-structured interviews.

In the context of phenomenological inquiry, it is often advised to use sample sizes which entail at minimum three participants (Dukes in Creswell, 1998, p. 122). For the current study, insight from interviews with six GZCS staff members, an electronic survey of ten participants to the case study, as well as literature review, informed the data collection and analysis process. While observations of all operational activities were conducted, recording of interactions between one project manager, one software engineer and one operations engineer were recorded for analysis.

Furthermore, even after implementation of a proposed information system prototype during the case study, observations of participants during their day to day routine was conducted to collect data which was largely qualitative in nature. This was to capture subject's experiences in their daily activities prior and post using the proposed technological tool. Analysing human gathered data, especially that which was qualitative in nature, was an additional factor in leveraging interpretivist perspectives where deemed appropriate.

4.2 Synopsis on positivism and interpretivism

Research objectives tend to determine the choice of epistemology to be adopted. For example, positivism is mostly used in studies where emphasis is placed on thorough replicable observations of phenomenon. In contrast, interpretative studies are mainly used to describe how such a phenomenon has emerged and appears (Trauth & Erickson 2012, p. 3).

Information Systems studies explore the way organisations can be best served through the usage of software built by engineers to improve day to day activities. Research in this field can be of a quantitative or qualitative nature. Understanding this aspect of Information Systems highlights why researchers may opt for different approaches to tackling their research questions depending on the objectives they wish to achieve.

In practice, Information Systems researchers have identified two main philosophical approaches of as point of interests for most research in the field: positivist paradigms on one hand and the constructivist and interpretative approaches on the other (Mwansa 2015, p. 48).

As mentioned earlier in the western history snapshot of epistemology, positivism has had a central influence in academic research. It is even equated to science itself in that it advances a certain absolutism in its conception of knowledge as the only objective truth (De Villiers 2012a, p. 223). Positivist researchers consider that academic enquiry should be conducted mainly with quantitative measures to remove bias and ensure consistency. Advocates of positivism believe that such studies should be driven by clearly enunciated hypothesis and their results should yield accurate scientific observations and predictions.

Interpretivism in terms of academic research places a bigger emphasis on the qualitative nature of knowledge and strives to find a deeper meaning in the underlying nature of the human experience. This paradigm considers that reality can only be understood subjectively in terms of human construction. Considering this, for interpretive studies such as the current inquiry which deal with human generated qualitative data, one is compelled to take into account the fact that a researcher cannot ignore the social context of the phenomenon he or she is attempting to study (Kroeze 2012, p. 47).

4.3 Hybrid IS research: blending interpretive and positivist/engineering perspectives

The Information Systems field of study deals with the usage of Information Technology to develop software systems. It also describes the contextual influences of environments as well as the interactions of organisations, groups and individuals with these systems (Sidorva *et al.* 2008, p. 475).

When positivist themed studies tend to adopt a single perspective on the essence of knowledge, interpretivism considers that research must be conducted by considering that there is a multitude of explanations to phenomenon. In terms of Information Systems research, the interpretivist paradigm accepts that researchers and study subjects should interact and hence influence each other to arrive at a more complete understanding and effectively meaningful knowledge (Kroeze 2012, p. 48).

The current research suggests a practical agile oriented development lens for delivering quality software. A key aspect to consider for this was to ensure that research results would be generic enough to be replicated, and that they would additionally be applicable to the South African context more specifically. An interpretative approach was hence adopted to cater for the local contextual industry realities African start-ups must deal with. Choosing the interpretative paradigm enabled the researcher to consider his own perspective and experiences in terms of software quality delivery culture. Having adopted a predominantly interpretative postulate, additional considerations still had to be considered.

This research opted to also cater for a positivist/engineering point of view. This is because in many cases, some IS studies have been observed to require a triangulation of methodologies ,epistemological approaches as well as data gathering strategies to cover a broad enough spectrum of pertinent knowledge areas (De Villiers 2012b, p. 239). When the current study proposes a "*Practical agile oriented framework*", it is considered that such a framework would gain insight by being informed by a technological tool developped to address some of the enunciated research propositions in Chapter One.

In the subsequent sections, Chapter Four proceeds to further discuss the research approach methodology utilised throughout this study.

4.4 Qualitative approach to Information Systems Research

Qualitative research enquiry processes are typically used to explore issues dealing with human or social problems within a certain context. Consequently, qualitative approaches usually put a special emphasis on the philosophical perspectives and assumptions of the researcher while taking into account the viewpoints of subjects being studied in the context of academic inquiry (Vaismoradi, Turunen & Bondas 2013, p. 399). This is one of the central reasons why readers need to consider researcher bias if they are to understand how certain conclusions come to be in a qualitative study context.

A common motivation for using qualitative methods is the fact that complex phenomenon exploration presents challenges when researchers opt to exclusively adopt quantitative measuring tools. This is most evident where propositions are set at an intersection between technical, non-technical, and project management issues (Dyb*å et al.* 2011, p. 426).

Additionally, qualitative approaches provide a large array of naturalistic and multi-method toolset to tackle research questions (Mor*a et al.* 2012, p. 224). This toolset of available options includes conducting direct observations of subjects, interviews of research participants, case studies to gain practical insight, thematic analysis of text material etc. (*ibid.* 2012). Qualitative researchers are also attracted by this methodology as they consider it to facilitate more realistic analysis of data items and hence help improve understanding of organisational settings, processes and real world scenarios (Akbar *et al.* 2014, sec. 2).

More often than not, qualitative studies aim to achieve an understanding of events by building holistic pictures and report detailed views of subjects in their natural operational setting (*ibid.* 2012).

In the context of Information Systems research, qualitative studies tend to formulate hypotheses informed by sound theory while considering at the same time the various complexity dimensions a research problem may infer. Such research helps IS scholars understand what goes on in real world settings by facilitating the observation of interactions between information systems, processes and people (Jabar et al. in Orso & Rothermel 2014, p. 50). Additionally, a key interest in adopting qualitative lenses when conducting IS research is the capacity to observe the effects of business and organizational change by information systems (*ibid.* 2014).

It seems pertinent to highlight a few cornerstones which informed the current thesis ‘research approach:

- A qualitative research perspective facilitates the observation of Information Systems effects and impact in organizations.
- Qualitative research findings are scientific enough that they can even be utilised as the departure point of rigorous quantitative studies (De Villiers 2012b, p. 239).

- Qualitative and quantitative methods are not always antithetic. It is for instance at times necessary to use data collection techniques acceptable in both quantitative and qualitative studies.

To adhere to the rigorous scientific standards required in any serious IS research thesis, a sound theoretical foundation is essential. In the next section, Activity theory which was the guiding theoretical framework of this research is discussed.

4.5 Theoretical Foundation

4.5.1 A brief introduction to Activity theory

The works of Vygotsky and Leontiev in the early 1920s contributed to the emergence of Activity Theory (AT) as a lens to facilitate the exploration of phenomena while improving the understanding of human activity (Hasan & Kazlauskas 2014, p. 9).

At inception, Activity Theory ambitioned to enhance the study of individual beings, the societies and groups within which they operate to better grasp what drives them and why it is they do the things they do in their everyday settings (Kaptelinin & Nardi 2007, p. 31).

It was hence used in social sciences in general and psychology specifically to help researchers further their insight on human behaviour in social environments.

By focussing on pedagogical practices in the context of developmental psychology, theoretical works around AT were often aimed at improving the skills potential dormant in individuals interacting through mediating tools within an activity system (Caroll, Bertelsen & Bødker 2003, p. 11).

Initially grounded in the field of human psychology, this theory has also been used in other fields including ethnography, human-computer interaction, information systems research, communications, education etc. (Karanasios, Riisla & Simeonova 2017, p. 1).

AT can be described as a conceptual framework which is centred around the concept of human activity and emphasises the precedence of human actions over cognitive models (Hasan & Kazlauskas 2014, p. 9). In recent times, researchers have used the Activity Theory as a design tool as well as a qualitative analytical framework in emerging fields such as Service Design for instance (Zahedi, Tessier & Hawey 2017, sec. 2.1).

Much of Activity Theory's appeal can also be found in Information Systems research and the software engineering field. This is because it proposes an attractive framework in these disciplines as by enabling the exploration of mediation in organisations through the usage of computing technologies which are observed to facilitate human activity (Baumer & Tomlinson 2011, p. 134).

Table 4-2 serves as an overview of traditionally discussed themes when studying the Activity Theory (Hardman 2005, p. 380):

Table 4-2: Activity Theory key notions as drawn from Russell (2002) & Cole (1996) in Hardman (2005)

HUMAN ACTIVITY	This is the study of human activities as set of collective individual actions. In Activity Theory studies, researchers consider that individuals only seemingly act in an isolated context. Rather, it is observed that variables such as group interactions, social and historical context all have an influence on individual actions.
TOOLS	This theme discusses the instruments which mediate human activities and impact their behaviour.
CONTEXT	This theme discusses how despite relative levels of agency, human behaviour is hugely influenced by the tools that are imposed upon them and thus constrain the limits of their actions.

Understanding the *Context* dimension of inquiry is important because in activity theory-based studies, emphasis is placed on the explanatory nature of phenomena. Consequently, researchers who ground studies in AT are encouraged to give ample space to interpretation (Hardman 2005, p. 281).

Activity theory could hence be deemed a nice fit for an interpretative study such as the current one as it has proven itself very useful for interpretivist researchers conducting similar qualitative inquiries. In practice, such studies require a clear grasp of how cultural as well as environmental factors influence the execution and management of processes and their supporting tools (Joshua, Nehemiah & Ernest 2015, p. 267). AT frameworks also put a huge emphasis on the importance of understanding a subject's motivations and objectives in organisational settings (*ibid.* 2015). Furthermore, it has also been demonstrated to be usefully as a broad theoretical framework in a variety of technology oriented studies (Kaptelinin & Nardi 2007, p. 97).

Activity theory researchers describe an entire work action as the main unit of analysis where an *object* as well as the *subject* and *tool* are used as analytical components (Hashim & Jones 2007, p. 5). In such activity system-based studies:

- The term *subject* describes the individual observed.
- *Object* refers to the aim(s), end goal(s) and objective(s).
- The concept of *tool* depicts mediating artefacts which enable the execution of tasks by subjects.

Recent expansions to the traditional activity theory framework have been advanced. Figure 4.1 illustrates Engestrom's expended activity theory model:

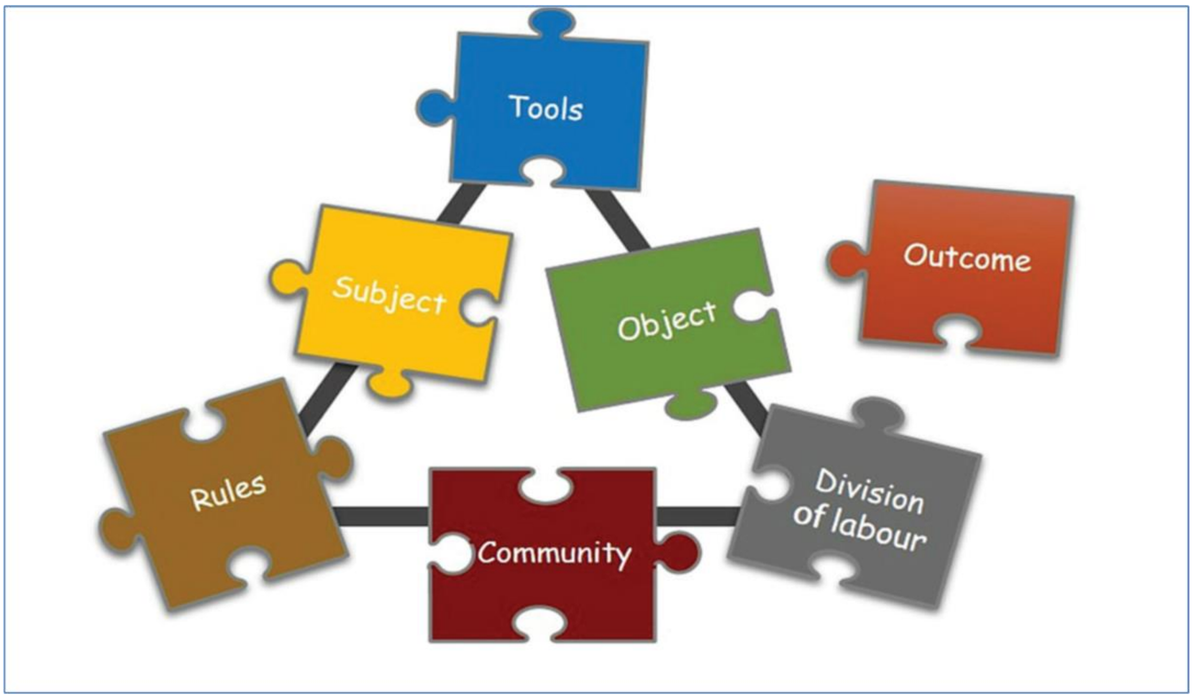


Figure 4.1: Activity theory model (Gedera & Williams 2016)

Grounded in Vygotsky's initial model, Engestrom introduced the concept of *rules* which encompass the conditions which influence human behaviour through social conditioning (Hashim & Jones 2007, p. 5). Another unit of analysis added in this activity theory model is the *division of labour* which describes the distribution of tasks amongst members of what is identified as the *community* (*ibid.* 2007). The following section discusses Activity Theory in the context of academic research with the aim of eventually describing how it was applied to this study's research objective.

4.5.2 Activity theory in research

Research projects underpinned by an activity theory lens suppose at least three key methodological requirements (Hasan & Kazlauskas 2014, p. 12):

- The identification of the main activities to be observed in conjunction with each activity's subject, object and outcome.
- The identification of actions which constitute these activities
  and their mediating tools.
- The description of interactions between and throughout the identified activities.

In this light, research which aims to conceptualise an activity system needs to observe and describe details of worker interactions, mediating tools and equipment used to execute activities, the governing rules of the work context as well as the ultimate outcomes expected from the community (Engestrom in Hashim & Jones 2007, p. 6). Considering all the above, the next section proceeds to describe how Activity theory underpinned this thesis.

4.5.3 Applying an Activity theory lenses in the current study

Predicated by an Activity Theory analytical framework, previous studies have described IS projects as a sum of collective work activities (Kekwaletswe & Gaoussou 2012, p. 61). In terms of software engineering research, Activity Theory has been successfully applied before. One such study was conducted at an American software delivery group where activity theory was used to identify software engineering as the central activity; the *outcome* expected with regards to the activity system was high quality software which is as free from bugs as possible (De Souza & Redmiles 2003, para. 6). Additionally, the *object* was identified as the end system being built (identified as *MVP Software* in the study) and the *community* described as the engineering team. Figure 4.2 describes how Engestrom' model was applied in this study (*ibid.* 2003):

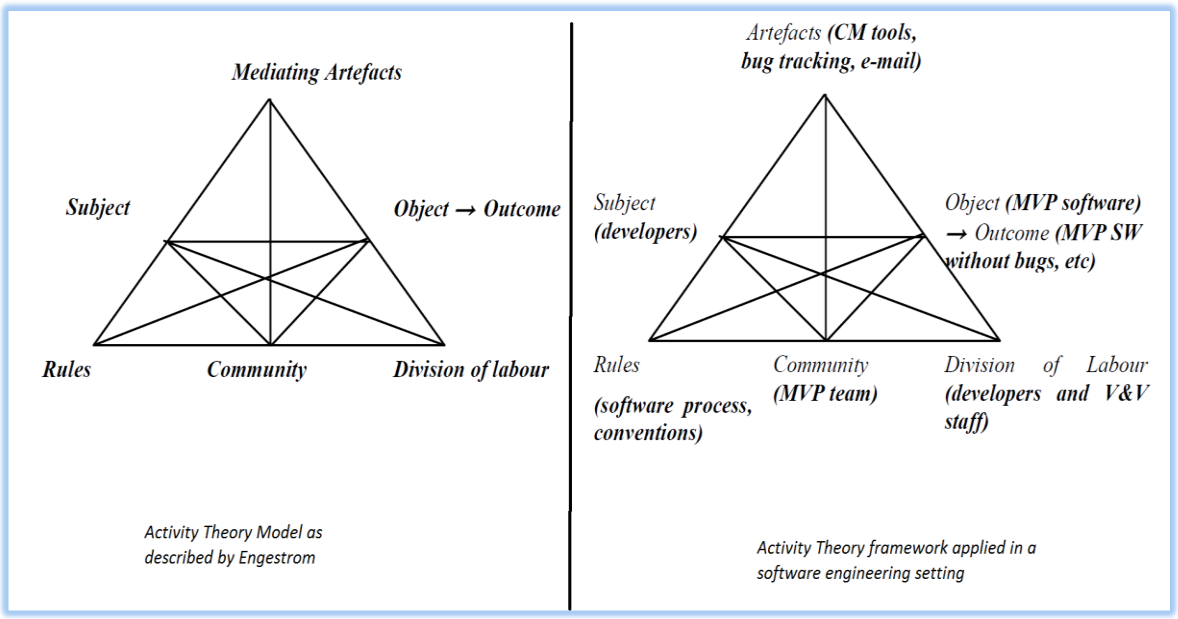

Mediating Artefacts
Subject
Object → Outcome
Rules
Community
Division of labour
Activity Theory Model as described by Engestrom
Artefacts (CM tools, bug tracking, e-mail)
Subject (developers)
Object (MVP software) → Outcome (MVP SW without bugs, etc)
Rules (software process, conventions)
Community (MVP team)
Division of Labour (developers and V&V staff)
Activity Theory framework applied in a software engineering setting

Figure 4.2: Application of Engestrom's enhanced model in a Software Engineering setting

Other studies have opted to apply Activity Theory in more traditional IS grounded themes. Inspired by research conducted in the educational field, local researchers have for instance implemented activity theory framed models on Learning Management Systems (LMS) adoption at a large South African parastatal (Gaoussou & Kekwaletswe in Iadis 2012, p. 282). This study identified *Instruction* and *Learning* as the central activities and a Learning Management system as the technology mediating tool (*ibid.* 2012). Figure 4.3 presents more details on the study's proposed activity theory inspired framework:

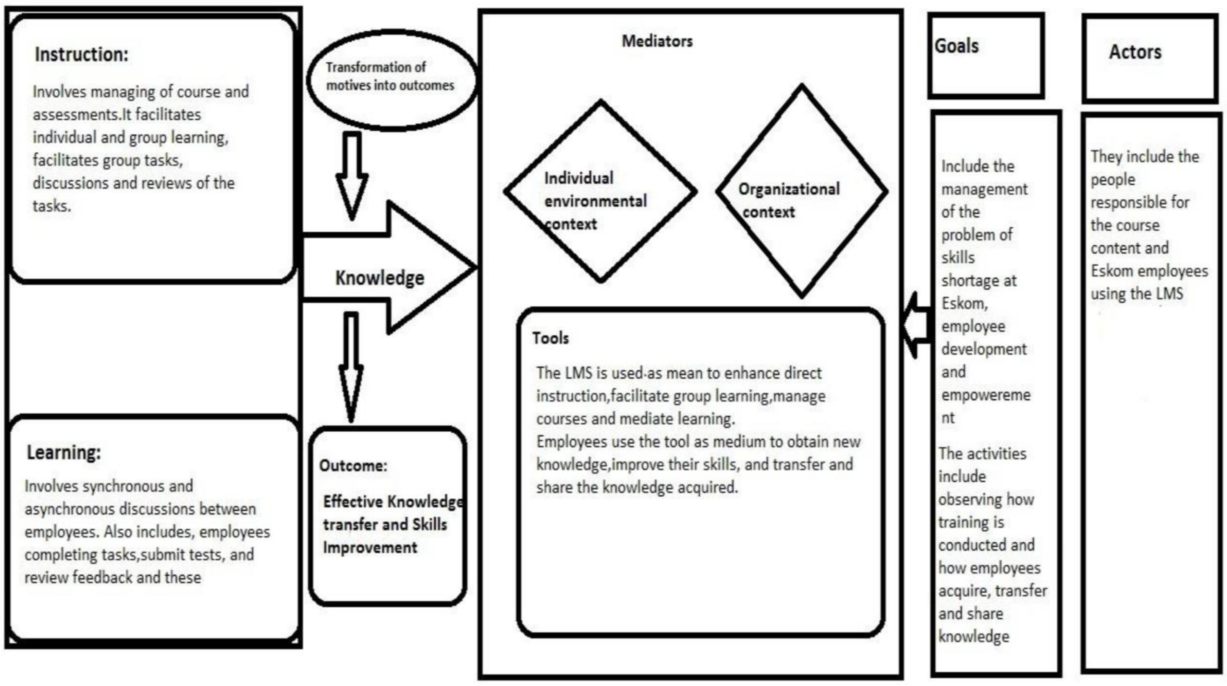

Instruction:
Involves managing of course and assessments.It facilitates individual and group learning, facilitates group tasks, discussions and reviews of the tasks.
Transformation of motives into outcomes
Knowledge
Learning:
Involves synchronous and asynchronous discussions between employees. Also includes, employees completing tasks,submit tests, and review feedback and these
Outcome:
Effective Knowledge transfer and Skills Improvement
Mediators
Individual environmental context
Organizational context
Tools
The LMS is used as mean to enhance direct instruction,facilitate group learning,manage courses and mediate learning.
Employees use the tool as medium to obtain new knowledge,improve their skills, and transfer and share the knowledge acquired.
Goals
Include the management of the problem of skills shortage at Eskom, employee development and empowerement
The activities include observing how training is conducted and how employees acquire, transfer and share knowledge
Actors
They include the people responsible for the course content and Eskom employees using the LMS

Figure 4.3: LMS activity theory-based model as described by Gaoussou & Kekwaletswe in IADIS 2012.

The Activity theory model is attractive to researchers in the IS field as it facilitates studies which cover a maximum amount of alternative contexts in terms of ubiquitous computing analysis (Achour et al. in Iadis 2012, p. 66). In some case studies, it has even been used for example to assist in improving requirements analysis during software development projects (Ahmad *et al.* 2013, p. 148).

The current study aims to produce an agile oriented framework supported by technology for the delivery of quality software. Activity theory was deemed a fitting theoretical framework which would consider specific environmental constraints unique to local markets. Additionally, an activity theory lens in the context of Information Systems research can be used to place technology as the central tool which mediates actions of individuals in the community (Hashim & Jones 2007, p. 6).

As the current thesis also ambitioned to produce a technological platform, an information system was built to consolidate information flows from various agile technologies and will be further presented throughout Chapters Five and Six. This information system in question forms the central mediating tool of the thesis' activity theory inspired approach. The tool itself does generate any new data to mine and its own internal database serves non-functional requirements which are transparent to users. It is built to provide a unified view of information which is amalgamated solely from existing data points generated by tools already in use at GZCS.

In effect, as mentioned in earlier sections, the current thesis draws from previous IS activity theory framed research. Informed by previous studies describing information systems as a mediating tool in an activity based system, this enquiry attempts to map research goals to an activity theory perspective (Hashim & Jones 2007, p. 6). In this light, the current study has initially identified:

- *Agile Software development or engineering as* ***Central Activity 1.***
- *Agile Software project management as* ***Central Activity 2.***
- *The local African start-up and software engineering industry as* ***External Mediator 1.***
- *GZ Consulting Services (local African software house also described as GZ CS)'s organizational culture as* ***External Mediator 2.***
- *The entire agile technology stack GZ CS uses to build its software for the African market as* ***Internal Mediators.***
- *An information system named "GZ-AgilePM Consolidator" as the principal technology* ***Tool*** *meant to facilitate the continuous delivery of quality software.*
- *The entire GZ CS team as the* ***Community*** *around which the activity system revolves.*
- *A consolidated information view of agile software project information for project managers and the entire software engineering team as* ***Outcome 1.***
- *The delivery of quality software as* ***Outcome 2.***

Having described the cornerstones of the current study's activity system, the next section proceeds to discuss how a practical case study was conducted at a local South African engineering start-up: GZ Consulting Services (pseudo name).

4.5.4 Using the Case Study as means of describing phenomenon

A case study from the interpretative perspective, can be described as an empirical enquiry engrained in the real world (Schwartzel & Eloff 2012, p. 278). This approach to conducting research is especially applicable when attempting to grasp behavioural implications to consider when one observes people and their systems in a natural setting (*ibid.* 2012). It also facilitates the exploration of modern organisational problematics in a realistic manner (Owoseni & Twinomurinzi 2018, p. 4).

In the fields of computer science in general and software engineering in particular, conducting case studies have been observed to be very suitable as they facilitate a researcher's enquiry into modern phenomena in natural settings (Runeson & Höst 2009, p. 131).

Case study approaches to exploring phenomenon are in similar fashion frequently used for qualitative IS studies (Jabar et al. in Orso & Rothermel 2014, p. 48). In Information Systems research, a case study could for example involve exploring a specific situation previously experienced at one or various organizations; the researcher would then identify a similar situation at a contemporary organization; such as a study could then proceed to a systematic comparison of the previous and current phenomenon while taking into account external environmental variables such as organisational culture, economic conjuncture etc.

In the current thesis, the researcher conducted a case study confined at a South African local start-up: GZ Consulting Services (GZCS). This firm is in a nutshell an agile software engineering house with the ambition to disrupt local markets by providing innovative retail as well as enterprise grade software solutions. GZCS is led by a managing director who handles both business strategy, commercial related endeavours as well as enterprise architecture. Reporting to him are Dev-Ops and Project/Engineering Managing functions. The team is additionally composed of four core systems engineers. An interesting point to note is that engineering tasks are shared at certain points and times throughout the organization. This means that the managing director volunteers and provides his time when required in assisting his team to tackle technical software engineering issues when required. Below is a snapshot of the firm's organizational structure:

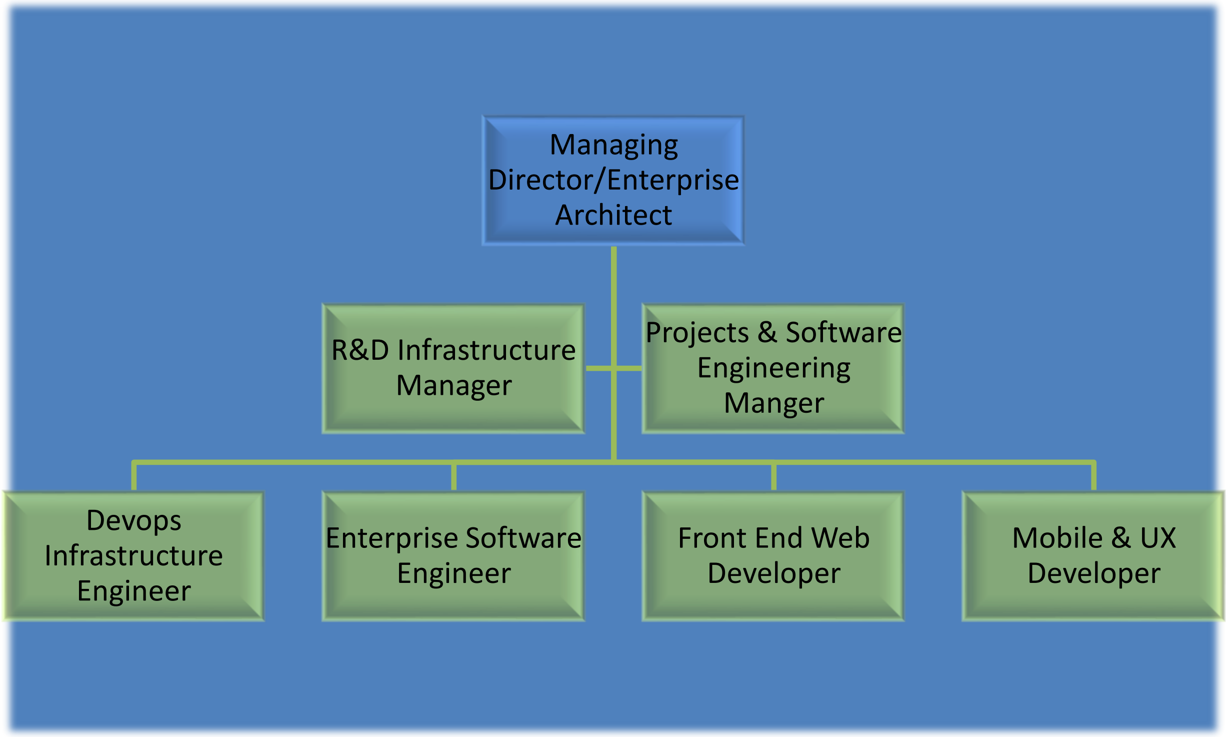

Managing Director/Enterprise Architect
R&D Infrastructure Manager
Projects & Software Engineering Manger
Devops Infrastructure Engineer
Enterprise Software Engineer
Front End Web Developer
Mobile & UX Developer

Figure 4.4: GZ CS Organisational Chart

This organization was in part chosen as it is a local African engineering house which places market disruption, quality and agility as its core values. Conducting a case study at this firm also seemed pertinent as it could help address key research objectives previously enunciated. These include:

- Getting an understanding of tailored engineering processes at a software organisation operating in South Africa.
- The identification of key agile technologies in use by such an organization to achieve software quality delivery.
- The implementation of an information system which will complement existing tools and facilitate the management of agile development projects and software quality delivery.

The next section of this chapter proceeds to a synopsis of data gathering and processing techniques employed for this study.

4.6 The data collection process

The incentive of gathering and analysing data from as many angles and perspectives possible often pushes researchers in adopting various data collection techniques. This can motivate research endeavours in the adoption of either qualitative or quantitative techniques. Depending on context, a combination of both techniques can also be leveraged to provide a variety of perspectives in the analysis phases. Another reason researchers often opt for a triangulation of quantitative and qualitative data collecting approaches is the fact that they are not necessarily antithetic (Mora *et al.* 2012, p. 224).

Nonetheless, some triangulation endeavours have shown limitations to drawing broad conclusions from information sets gathered can using different techniques. Although this can happen if the datasets analysed are not actually comparable, in practice, quantitative and qualitative data collection methods for instance can be complementary. Furthermore, triangulation is mostly associated with the mixing of research designs as well as their associated implementation techniques (Heale & Forbes 2013, p. 98)

Figure 4.5 describes the variety of options available to collect data in positivist or interpretivist styled studies.

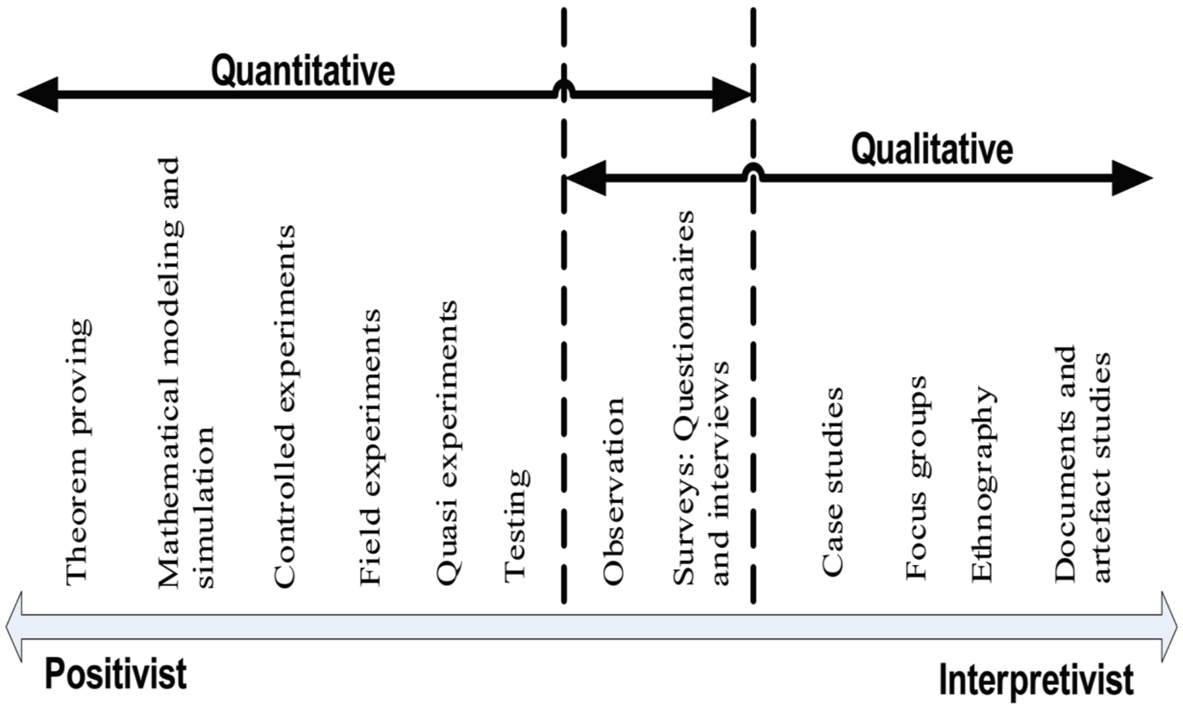


Figure 4.5: Research Methods (De Villiers in *Mora et al*. 2012, fig. 1)

In this thesis, software development and software project management in an agile manner were identified as the central activities. These two elements also form the base of the activity-based system where and agile project management information consolidator system is argued to be an enabling tool for quality software delivery.

In this light, the data collection process was informed by insights gathered from literature reviewed in chapters two and three. This was done by gaining a better understanding of historical and current trends regarding engineering methodology, agile practices, tools, and related topics.

To further this insight, the researcher opted to use semi-structured interviews, direct observations and questionnaires as data collection methods during the case study. These data collection techniques are briefly discussed in the next sections.

4.6.1 Semi-structured interviews

Interviews typically entail having one-on-one exchanges between the researcher and individuals studied. The aim is to retrieve information relevant to the research objectives. This data collection method can yield pertinent information collected from participants which will provide insight into their beliefs, practices and opinions (Harrell & Bradley 2009, p. 24).

Interviews conducted in a semi-structured manner draw from structured and unstructured formats. With these types of interactions, the interviewer identifies key discussion points to be addressed (Harrell & Bradley 2009, p.25). The researcher hence follows the standards in guiding his questions formulation although he or she has leeway in the sequencing of questions. Some questions are usually open-ended and most of the time centred around predetermined themes while others are more spontaneous and emerge from the dynamic interactions between subjects and the researcher (DiCicco-Bloom & Crabtree 2006, p. 315).

These types of interviews are somewhat conversational and are typically used when one wants to examine a topic thoroughly and understand respondents’ answers to the issues discussed.

In the current research exercise, semi-structured interviews were conducted with individual software engineers and management staff at GZCS after the aim of the study was clearly explained to them. The decision to use semi-structured interviews was taken to create a free-flowing exchange with staff while retrieving their invaluable feedback and input in a manner that was scientifically rigorous without creating an atmosphere that could have been interpreted as too formal. The aim was to get a feel for what staff understood as quality in their line of business and how they believed their day to day activities could be enhanced in terms of quality software delivery.

4.6.2 Direct Observations

Harrell and Bradley (2009) suggests that during observations, the researcher should not participate in the interaction. Using this method, researchers aim to simply observe participants executing their usual activities. This occurs in their natural environment although the researcher is free to set or remove certain environmental variables relevant or not to the study. One should note that this collection method does not necessarily yield completely objective data results as the mere presence of an unusual external element (researcher presence for instance) may influence participants into modifying their behaviour or not. This limitation must be understood, and observations alone are hence not enough without considering organisational or environmental context when interpreting the data collected. Direct observations of software engineers and project management staff were conducted prior and post the proposed agile oriented information systems implementation in their routine project activities. Central activities 1 (Agile development) and 2 (Project management) are the main activities that were deemed relevant to observe. Collecting data in this manner was mostly conducted between August and December 2017.

4.6.3 Questionnaires

Questionnaires are designed as a set of questions submitted to study subjects with the aim of getting answers typically confined within research objectives boundaries (Sekaran 2003, p. 236). This is a widely used data collection method in qualitative research when the researcher is interested in getting subjective insight from participants (*ibid.* 2003). Questionnaires were provided to current GZCS staff to get views of management and engineering staff alike. All questionnaires were completed, returned for analysis by the researcher. Limesurvey was the electronic tool used to implement and distribute this study's questionnaire to participants.

4.7 The data analysis process

The bulk of data gathered during this thesis was qualitative. In order to identify major themes which emerged during literature review, the research adopted a content as well as thematic approach to analyse data. In addition to this, the activity theory framework was also used as an analytical lens.

4.7.1 Thematic analysis

Thematic analysis is commonly used to examine collected data in qualitative research (Guest et al. in Javadi & Zarea 2016, p. 34). It involves rigorous examination and exploration of data with the aim of extracting relevant themes from the phenomena being studied (Fereday & Muir-

Cochrane 2006, p. 82). Previous studies have shown this analysis approach to be especially efficient when attempting to describe textual data's explicit or inferred complexities (Blayney & Harreveld in Dervin *et al.* 2016, p. 210).

In the current study, participants are experienced and knowledgeable technology and project management professionals. Thematic analysis has been shown to be appropriate in exploring data collected by expert knowledge workers as it facilitates the identification and reporting of patterns (Suhairi *et al.* 2017, p. 1315). In conducting this research, a qualitative questionnaire was implemented and distributed to participants. As key themes categorized individually emerged from data collected during the questionnaire, thematic analysis was used to better understand this information. Furthermore, Activity theory was applied as the principal theoretical lens to gain deeper insight on participants' answers.

4.7.2 Content analysis

When conducting content analysis, researchers examine and methodically analyse qualitative data (Sekaran 2003, p. 410). Traditionally, content analysis was considered quantitative in nature as it enables researchers to draw quantitative inferences by submitting findings to statistical analysis (*ibid.* 2003). It has proven to be an attractive analysis lens for qualitative research as well in more recent times. In the context of qualitative research, this technique advocates the scrutiny of text sources to identify meaningful patterns and gain further insight (Hsieh & Shannon in Hashemnezhad 2015, p. 59). It can complement thematic analysis by expanding on the description and interpretation of extracted data sets.

Content analysis was hence applied to documentation explored during the case study as well as interview transcripts and analysis. QDA Miner, MaxQDA and NVivo were the software suites used by the researcher to perform computer assisted analysis of core textual data.

4.8 Research Validity

To ensure the scientific nature of an inquiry, researchers need to convince readers that data was appropriately collected and that the manner in which it is evaluated accurately gages the described phenomenon (LeCompte & Goetz 1982, p. 32).

For research conducted in most disciplines, the more rigorous and demonstrative the data presentation and analysis appears, the more valid the study is often perceived. For cultural-historical works, this validity can even be increased when one takes into account that the subjective postulate of a researcher will always influence somewhat the gathering and dissection of data (Gedera & Williams 2016, p. 11).

In the context of qualitative oriented studies, some have suggested that approaching the issue from an angle focussed on rigor is more appropriate than the traditional quantitative perception of reliability and validity (Kelle & Laurie in Welsh 2002, pt. 4). For this thesis, although electronic questionnaires were used as one of the information collecting techniques, the instrument used did not contain Likert scale interrogations. This is because the researcher aimed at gathering descriptive responses from participants. Although this is was an arbitrary consideration as can be tolerated with an interpretivist positioning, the researcher experienced that when using questions formulated in rating scale formats, getting target audiences comprised of engineers to expound on their answers can be challenging. The direct consequence of not opting to use rating scales coupled with the small participant sample of the research case study, resulted in the inapplicability of metrics such as Cronbach's alpha.

To analyse data retrieved during interviews or questionnaires, leveraging technology with software analysis tools could hence be considered as an efficient mean to add robustness to a study. This can an efficient means of supporting appropriateness of the methodology adopted if one focuses its usage around clearly enunciated research questions.

The appropriate application of such software can add rigour to a thesis such as the current one and perhaps increase its perceived relevance (Richards & Richards in Welsh 2002, pt. 4). Sophisticated qualitative tools such as the NVivo software suite were in this light also used to dissect the qualitative data for this thesis. During the current research exercise, for the case study conducted at GZ Consulting Services, NVivo, MaxQDA and QDA Miner were the primary software suites used to gain more insight most of the qualitative data collected.

As mentioned earlier, a researcher's initial postulate tends to influence and shape the research itself. Moreover, in the explorative journey of phenomena, especially in qualitative works, the researcher's views may also be influenced or altered (Palaganas *et al.* 2017, p. 426). As such, leveraging a framework like Activity Theory (AT) as was done in this thesis, enhances the capacity for the researcher to adopt a reflexive postulate to improve his comprehension of the phenomena at hand (Gedera & Williams 2016, p. 11).

To improve the robustness of data collection, analysis and interpretation, an AT lenses was hence primarily used to align the theoretical foundation discussed earlier in this chapter with retrieved data during the case study.

4.9 Chapter Four summary

The fourth chapter of this study introduced the research approach and methodology. Interpretivist and qualitative research elements were discussed to describe how the current dissertation was structured. Chapter Four also explained that the case study component of this thesis was conducted at South African local start-up GZCS. Data gathering and analysis techniques were then discussed. In this chapter, the thesis proceeds to introduce an information system designed and implemented during the ‘case study component of this research. Research findings are then presented and described in further detail during analysis of the information collected.

5. CHAPTER FIVE: DESIGN OF AN AGILE INFORMATION SYSTEM

As discussed in Chapter Four, this study is grounded in the foundations of Activity Theory. The current chapter hence introduces the activity theory oriented conceptual framework driving this thesis. Chapter Five then proceeds to further explain a rationale behind this case study to understand how the agile information system was designed.

5.1 Conceptual framework

During literature review, concepts such as software quality and development processes were explored. It was also suggested that an organisation could gain immense benefits by adopting agile approaches in running its software projects (Scheid 2013, para.7).

To tie this in with a practical mapping of Activity Theory for purposes of drawing an informed conceptual model, previous studies have developed an *Eight-Step-Model* (Mwanza-Simwami, 2001, sec. 5.1). This model can be used to understand the activity system being studied by identifying the following pillars:

Table 5-1 Eight-Step-Model (*Adapted from* Mwanza-Simwami, 2001, sec. 5.1)

Eight-Step-Model Element	**Description**
Activity of interest	*Set of activities which are to be studied.*
Objective of said activity	*The reasons for which activities need to be executed.*
Subjects in said activity	*The actors which execute required activities.*
Tools	*The instruments leveraged by participants to execute their activities.*
Rules mediating exercise of activity	*The possible mediating cultural factors or explicit governing directives.*
Division of labour mediating the activity	*The distribution of effort amongst actors.*
Community within which activity is executed	*The ensemble of actors.*
Desired Outcome of Activity	*The end goal of the activity.*

Informed by this model, the current research could draw a conceptual lens which would clearly explicit the activities and relevant elements to be observed. Furthermore, an activity theory centred study needs to conceptualise a framework which details worker interactions, mediating tools involved in day to day activities as well as contextual factors influencing outcomes expected from the community (Engestrom in Hashim & Jones 2007, p.6).

At this stage, the elements to be observed from the case study's activity system could hence be summarized as follows:

- The South African local start-up scene and organisational culture would be considered **Mediators** influencing project stakeholders and subsequent project activities.
- Project managers and engineers as **Actors** who form of the **Community** and execute their day to day activities. This community distributes work tasks through an implicit **Division of Labour** depending on whether staff members are engineers, responsible for project managing functions. Their **Objective** is to effectively execute their functions daily in the context of software engineering responsibilities and related project management tasks.
- Software Engineering, Dev-Ops Engineering and Project management are the key **Activities** which are performed by members of the community described earlier. These activities are executed with the help of technology.
- Technology including project management software, productivity tooling as well as cloud resources are the key **Tools** leveraged to conduct activities.
- Quality Software solutions effectively being delivered to market constitute the **Outcome.**

Figure 5.1 illustrates the consequent mapping of this model for purposes of the current study:

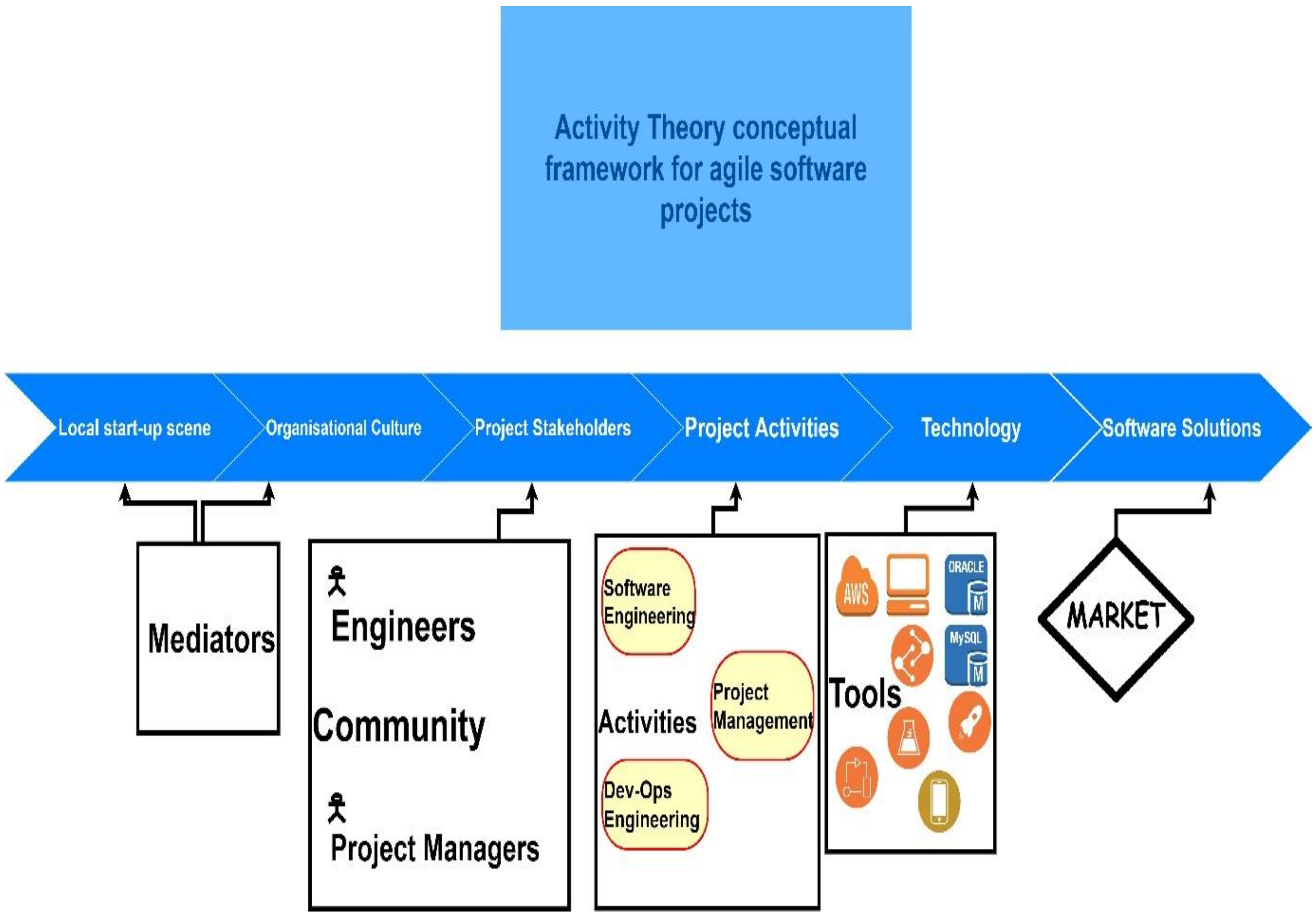


Figure 5.1: Activity System conceptualised framework for agile software projects

5.2 Refining the model to fit key research objectives

Amongst other aspects, figure 5.1 depicts a technology element of the conceptual framework which relies on a multitude of tools to deliver quality software solutions to markets. These tools comprise the bulk of technology systems utilised by engineers and project managers to conduct their daily activities. Drilling into this conceptual framework described earlier, from a technology perspective, one can attempt to further align this model with the previously stated research objective aimed at facilitating the delivery of quality software.

After analysis of internal company documentation during the current case study, it emerged that GZ Consulting Services leverages a multitude of technologies including cloud services to conduct its operations (Mugeni 2015a, p. 6). These technologies included Continuous Integration and Deployment software stack such as Jenkins, TeamCity, software development stack including Microsoft.Net and Java and cloud services such as Amazon Web Services etc. (*ibid.* 2015).

GZ Consulting Services is an agile software engineering house. Often, such modern organisations also adopt DevOps as well as Continuous Integration practices. When this is the case, teams can experience difficulties in managing progress and identifying components of software project that are affected by changes (Palihawadana *et al.* 2017, p. 130).

During the current case study, staff were questioned about which sources they used to inform their decision making and daily activities. The aim of these interrogations was to understand how team members performing different functions across varying domains were keeping track of colleagues' activities particularly when this affected their own functions. It emerged that team members were concerned only with technology stacks they were familiar with and used daily. Staff members themselves considered this posed a risk as the most accurate source of truth was mostly only available during stand-up meetings. From this observation, the proposed activity theory inspired conceptual model was refined so that it could consider risks associated with technology usage in the context of software projects:

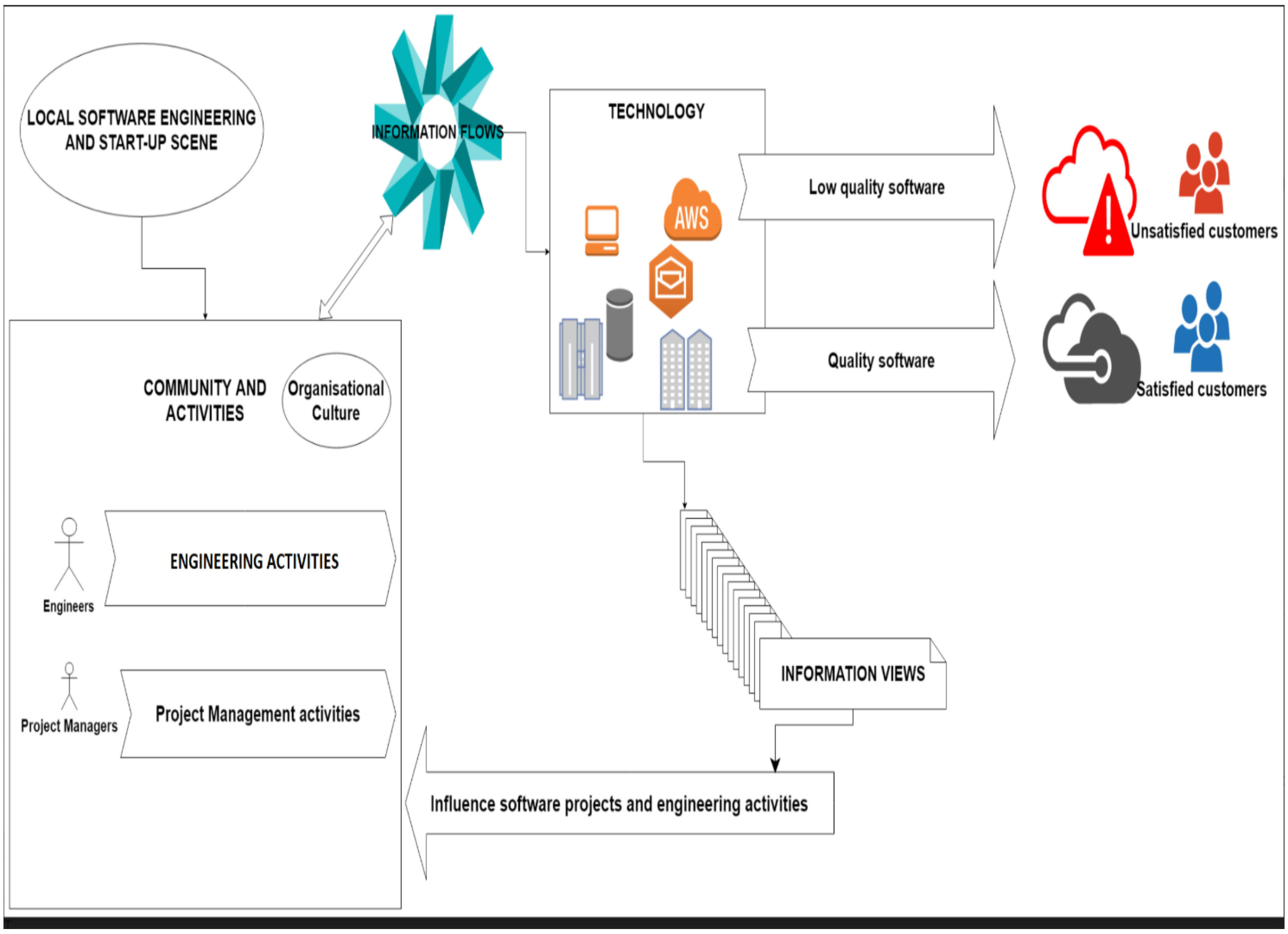


Figure 5.2: Conceptual framework with technology risks

In figure 5.2, emphasis is put on a situation where the usage of innovative and modern technologies such as productivity tools, modern databases or cloud products including the ones provided by Amazon Web Services does not guarantee the delivery of quality software and consequently customer satisfaction (Mugeni 2015a, p. 6). In practice, while technology can play a central role as the main tool in an activity system, each stack can yield an immense amount of information views which would in turn negatively influence project activities (*ibid.* 2015).

To address the issue of facilitating quality software project delivery in this context, an option would be to consolidate information views into a single source of consistent truth. This is what the researcher aimed to achieve with the implementation of an information system to deal with consolidation of information during the case study at GZ Consulting Services. In the next section, Chapter Five proceeds to a deep-dive discussion on the case study.

5.3 Presenting the case study

Influenced by an activity theory-oriented lens, the researcher implemented an agile information system to aggregate engineering and project management data for the case study at GZ Consulting Services.

This organisation was identified as an ideal subject for the case study at hand due to a couple of reasons. The company as mentioned before is a South African SME located in Johannesburg, South Africa. It is a relatively young small sized start-up with fewer than 20 employees. Because of this small size, it was considered that the company afforded the researcher an ideal context to perform concise observations and test the proposed prototype system. The fact that the start-up leverages agile methodologies and is an avid Dev-Ops advocate was an additional criterion of attractiveness.

The system implemented during this study ambitioned to provide a consolidated source of truth for project stakeholders. This experimental component of the thesis involved providing GZCS software team members as well as management staff access to the tool developed and letting them use it as they pleased after a basic on-boarding process.

In parallel, the researcher also conducted direct observations of GZCS staff during project development cycles as well as unstructured interviews to put participants at ease. An electronic survey was additionally utilized to gather staff member views regarding software quality, engineering and project management related concept prior and post the implementation of the researcher’s information system.

As a central component of data gathering in this thesis, it is perhaps pertinent to re-state the first two key aims of the case study as described in previous chapters:

- Getting an understanding of tailored engineering processes at a software organisation operating in South Africa.
- The identification of key agile technologies in use by such an organization to achieve software quality delivery.

Insight gathered from a conversation with GZ Consulting‘s managing director informs the next section where the firm’s technological landscape as well as its engineering processes are summarized.

5.3.1 GZ Consulting’s development philosophy and technological landscape

GZ Consulting services is a young emerging agile and Dev-Ops oriented organisation with staff located in Gauteng, South Africa. Internally, regarding its operating model, the company is an advocate of agile methodologies and practices. Projects are managed mostly through the establishment of sprints, Kanban boards, epics and user stories. The organization doesn’t adopt a dogmatic approach to its agile practices and customizes processes as much as it can to fit its specific needs.

GZ Consulting services is also engrained in a Dev-Ops context where both infrastructure and development staff are expected to be technically apt enough to be able to cross skill. Software engineers are required to be able to perform certain operational tasks such as deploying code, writing scripts etc. and infrastructure operators are expected to be able to contribute code when required (Mugeni 2015b, p. 7).

Regarding expenses, sub-Saharan markets the firm targets do not afford it the luxury of heavy footprint hardware and software costs. The management team hence considers that there is a need to adopt a modern and lightweight infrastructure footprint internally while proposing the same to its clients where appropriate.

The organisation has opted to run the entirety of its technology stack on cloud infrastructure. GZ CS has to date successfully deployed its products to the cloud without observing any notable differences between applications hosted on site or at third party premises (Mugeni 2015a, p. 3).

The results of this move to the cloud has produced an increase in team productivity while leveraging cutting edge technical architecture and infrastructure.

Senior management at the firm described that while there was an urgent need for cost effective infrastructure, it believed that adopting a coherent long-term cloud strategy would facilitate its growth objectives as well. Leadership at GZCS considers that embracing an enterprise-wide cloud strategy even in the context of its small start-up operating model presents two key opportunities. On one hand the ambition is to establish a structured way to incorporate cloud services into its internal IT mix. On another hand the firm wants to propose the same level of efficient products and services to its clients. The long-term vision for management is that if the firm's product and services offering is supported by a solid infrastructure foundation, GZ Consulting services is hoping to eventually emerge as a cloud-oriented centre of excellence in the delivery of software products and services on the local market as well as internationally.

Currently, GZ Consulting services prides itself in the fact that the question of how to leverage cloud technologies has been largely answered. Every software project at the firm is hosted on the GIT source control management system. The firm's intellectual property is hosted on private GITHUB cloud repositories which contain source code for the firm's web application and mobile devices services. These cloud repositories also host the firm's enterprise grade solutions which includes middleware software adapters as well as public and private application programming interfaces (APIs).

Project managers leverage a private Atlassian JIRA cloud instance to run internal as well as commercial projects. This suite helps leadership and executives track the management of software and business projects in an efficient ubiquitous manner. Support issues, bug reporting, enterprise strategy as well as architecture are functions also offered by the Atlassian JIRA cloud which the organisations utilizes.

From an engineering perspective, the entire technology stack in use at the firm leverages Amazon cloud infrastructure. Amazon Web Services (AWS) is used to host runtimes such as Java enterprise grade products, web applications built in NodeJS etc. Other service providers including Microsoft Azure' container services as well as Digital Ocean's offering have been previewed and tested by engineers at GZCS, but Amazon's AWS platform is the preferred option for development and infrastructure staff.

In terms of more novel and game changing technologies adopted in the cloud space, GZCS engineering teams also use innovative serverless approaches to designing software solutions by leveraging AWS's lambda. This serverless approach to hosting web service is leveraged by the team to execute shorter running processes.

Furthermore, to facilitate builds, automated testing as well as to enable deployments and delivery, GZCS has adopted the continuous integration (CI) paradigm. Practically speaking, a Jenkins instance for some projects and AWSCodeBuild for others are hooked into the source control system and executes test cases implemented by development staff after handling the building of new software artefacts. The same CI servers handles the deployment of new products to required environments including Development and Production servers. Table 5-1 illustrates a synopsis of the technology stack in use at GZ Consulting Services:

Table 5-2: GZ Consulting Services technology matrix synopsis

Software Framework Stack	**Infrastructure & Project Management**	**Continuous Integration**
Microsoft .net Java Standalone edition Java Enterprise Edition NodeJS Glassfish Sever versions 3 to 5 Wildfly server 10 Android Spring Framework MYSQL MSSQL OpenESB MuleSoft	Amazon EC2 Linux Amazon EC2 Windows Server 2016 Jira (from Atlassian) GitHub corporate organisation instance hosting 30+ code projects and products	Jenkins (running on AWS EC2) AWSCodeBuild

The list above is non-exhaustive, and the organisation is always open to new tooling as well as technology if this can increase productivity and improve the management of its projects.

The sheer amount of technology tools utilised at the firm has led to the generation of an immense amount of data. Project leaders and engineers alike need to take this into account to effectively run their projects. Nonetheless, both project leadership as well as engineering staff have indicated that this heavy technology usage can present drawbacks. In effect staff have bemoaned the fact that despite technology making their day to day tasks easier, at times, there seems to be a need to access too many different systems to get information sets that are often related and at times redundant. This can prove problematic in fast paced environments where information accuracy is critical during project lifecycles.

For example, project managers in their case would like to have more practical insight into feedback from engineers. This is because task updates from technical staff do not always convince project leaders. Management would hence like to be able to monitor practical development progress without having to know the intricacies of navigating through the command line of a GIT system for instance. In the next section, a discussion on how the large variety of information technology tools utilised at the firm and the data they generate motivated the case for building a system which would present a unified view of all this data follows.

5.3.2 Motivating factors for an agile oriented information consolidator

When running agile projects, organisations often experience challenges in keeping workloads integrated among team members (Stillwell & Coutinho 2015, p. 1). This can be accentuated in situations where the level of information available to team members is either incomplete or perhaps inexact. Another factor that can worsen situations is that agile environments have often yielded a lack of traceability management and documentation of extremely poor quality (Palihawadana *et al.* 2017, p. 130). In the case of engineers for example, there is an additional need to provide updated and consistent environment information to ensure smooth running of engineering tasks (ibid., 2017).

When an organisation decides to adopt new methodologies and technology tools, often, this initially has implications in regards to reduced efficiency resulting in increased overhead costs (Logigear 2013, para. 2). Moreover, the utilisation by project teams of the latest technology trends or tools does not guarantee the delivery of quality software projects and solutions (Mugeni 2015a, p. 6). Also, as discussed in section 5.2 of this chapter, feedback from GZ CS employees suggests that there could be benefits to having a consolidated and consistent version of truth in the context of agile projects for all stakeholders.

This thesis's case study proposed an information system which will complement existing tools and facilitate the management of agile development projects and quality software delivery.

Figure 5.3 illustrates a layered ArchiMate annotation model which describes key motivators for development of the project management tool to consolidate agile informed data at GZ Consulting Services:

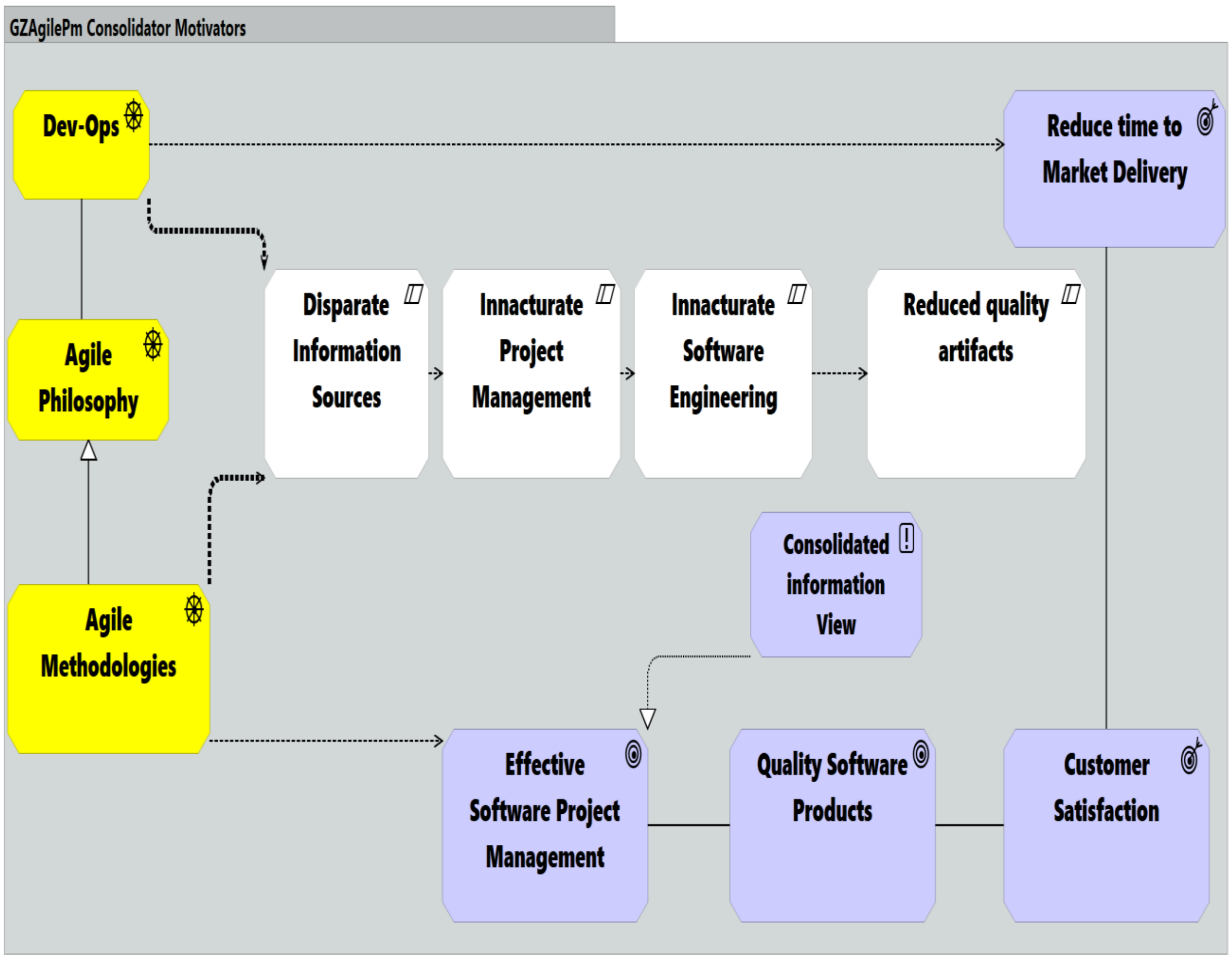

GZAgilePm Consolidator Motivators
Dev-Ops
Reduce time to Market Delivery
Agile Philosophy
Disparate Information Sources
Innacturate Project Management
Innacturate Software Engineering
Reduced quality artifacts
Consolidated information View
Agile Methodologies
Effective Software Project Management
Quality Software Products
Customer Satisfaction

Figure 5.3 Motivators for a consolidated information view

At GZ Consulting Services, the adoption of agile methodologies and Dev-Ops has been followed by extensive usage of supporting technology tools for both engineers as well as project managers. The sheer amount of information generated by these tools can be overwhelming. Furthermore, the disparate nature of information systems in use throughout development lifecycles can often lead to inaccurate project management (in terms of milestone appreciation, effort, cost and time estimations etc.). This could foster ill-informed software engineering projects which in turn can yield poor quality products. These factors have largely motivated the conceptualisation and implementation of an information system which would provide a consolidated information view for effective and accurate agile project management and thus potentially improve the quality of software products and perhaps increase customer satisfaction.

In the next section, an application and technology overview of the IS tool developed for the experimental part of this dissertation is further discussed.

5.4 Prototyping GZ-Agile Project Management Consolidator

As highlighted during previous discussions, even within the context of organisations adopting an agile philosophy and related methodologies, the large amount of disparate information sources can lead to inaccurate and incomplete assumptions in terms of project management estimating for instance, and engineering activities. This risk could exponentially increase as the array of agile practices and technology tools to support engineering projects grows.

Regarding the case study at GZ Consulting services, the researcher embarked on the design of an information system to aid engineering projects stakeholders by addressing the issue of information consolidation. The tool built is referred to as "***GZ-Agile Project Management Consolidator***" or GZAPMC.

In systems engineering, the *Open-Closed* principle suggests that a class should permit extensions to an object's behaviours but restrict modifications to its core (Dalling 2009, para. 1). The Open-Closed principle was applied from an architectural perspective in a limited capacity, to only complement current tools in usage at GZCS in its running of agile projects. The objective of this system would be to complement existing technology tools, facilitate the management of agile development projects and thus facilitate the delivery of quality software by providing a unified portal and view of relevant software engineering and project management activities**.**

To describe the information system implemented, subsequent sections hence proceed to schematically illustrate the business, application and technology views considered for development of this IS.

5.4.1 GZ-Agile Project Management Consolidator: Business View

In Aristotelian metaphysics, to understand an artefact's true nature, one needs to look at and consider its material, efficient, formal and final causes (Miller 1998, pt. 2). This involves understanding the object's content, its creator or process by which it came to be, its ideal form or shape as well as its actual function. One could leverage this perspective in understanding how IS tools should come about by identifying in project early stages details related to the system's requirements and expected functionalities.

In the current research setting, before discussing any technical design details of the IS prototype at the centre of this thesis' case study, the researcher identified a couple of aspects which required clarification before any development task could be started:

- Who would the system users be?
- What pre-requisite information would be required, and which medium would be provided to interact with the system?
- What would the core functionality of the system entail?

The system designed was conceptualised for academic research purposes, but the possible scope of application was deemed extensible to potential commercial applications in real life business settings.

Typically, Information Systems need to cater for administrative tasks and some level of business reporting capability. As such, the Agile Project Management Consolidator system caters for a Systems Administrator and Executive role although these users were considered accessory for the purposes of this study. The Administrator is meant to manage other user roles, load core non-business data into the system etc. The Executive user is meant to eventually be able to access reports and financial data. The functionalities required by these users were not deemed essential for the purposes of the thesis at this stage, so their interactions with the system are only superficial.

This study is informed by previous activity theory framed research (Kekwaletswe & Gaoussou 2012, p. 61). The information systems implemented at GZ Consulting services focuses as mentioned before on two central activities:

- Agile Software Development/Engineering (this includes activities considered in the DevOps realm).
- Agile Software Project Management.

Furthermore, as mentioned in GZ Consulting Services' technology matrix summary (table 5-2), the firm already leverages proprietary cloud software services to manage both its project management needs in the agile space, and its engineering requirements. It was hence deemed fitting that the solution to be built for prototyping purposes during this research, would be cloud based, and leverage the existing underlying infrastructure and software services already in use at the firm.

As such, the two core business actors considered during implementation of the Agile PM consolidator system were project managers and software engineers at GZ Consulting Services.

After the main intended audience of the Information Systems was decided, the researcher had to clearly identify what the tool implemented would require in complementing other technologies and products already in use at GZ Consulting Services. The researcher went on to design a system flexible enough to be able to cater principally for interactions through a web portal but also via mobile channels and Chat-Ops mediums such as Slack used at GZ Consulting services.

For the information systems to provide intended functionality to users such as retrieving data from the tools already in use at GZ Consulting Services, a credentials registration module is required. This functionality is used to secure internal users' data. It is also used to leverage essential third-party credentials, which enables the system to authenticate users to cloud services and backend systems they require information from. Once this is done, project managers and software engineers have access to a consolidated view of software project information aggregated from various sources used at GZ Consulting services. Additionally, users can initiate a complete project creation workflow which will synchronise and integrate the custom-built information system's artefacts to tools already in use at the organisation and hence monitor them as well.

To design a system which would address the problem illustrated in figure 5.3, the researcher considered insight from the firm's technical lead, who explained that although an agile advocate, the organization believed in sound architecture principles. In practice, solutions were designed to meet TOGAF principles whenever possible. Consequently, because data which to be consolidated from existing systems was not to be altered in anyway, the researcher opted to implement a proposed solution by incorporating three key TOGAF viewpoints: the business, application and technology views. Figure 5.4 illustrates the considered business view:

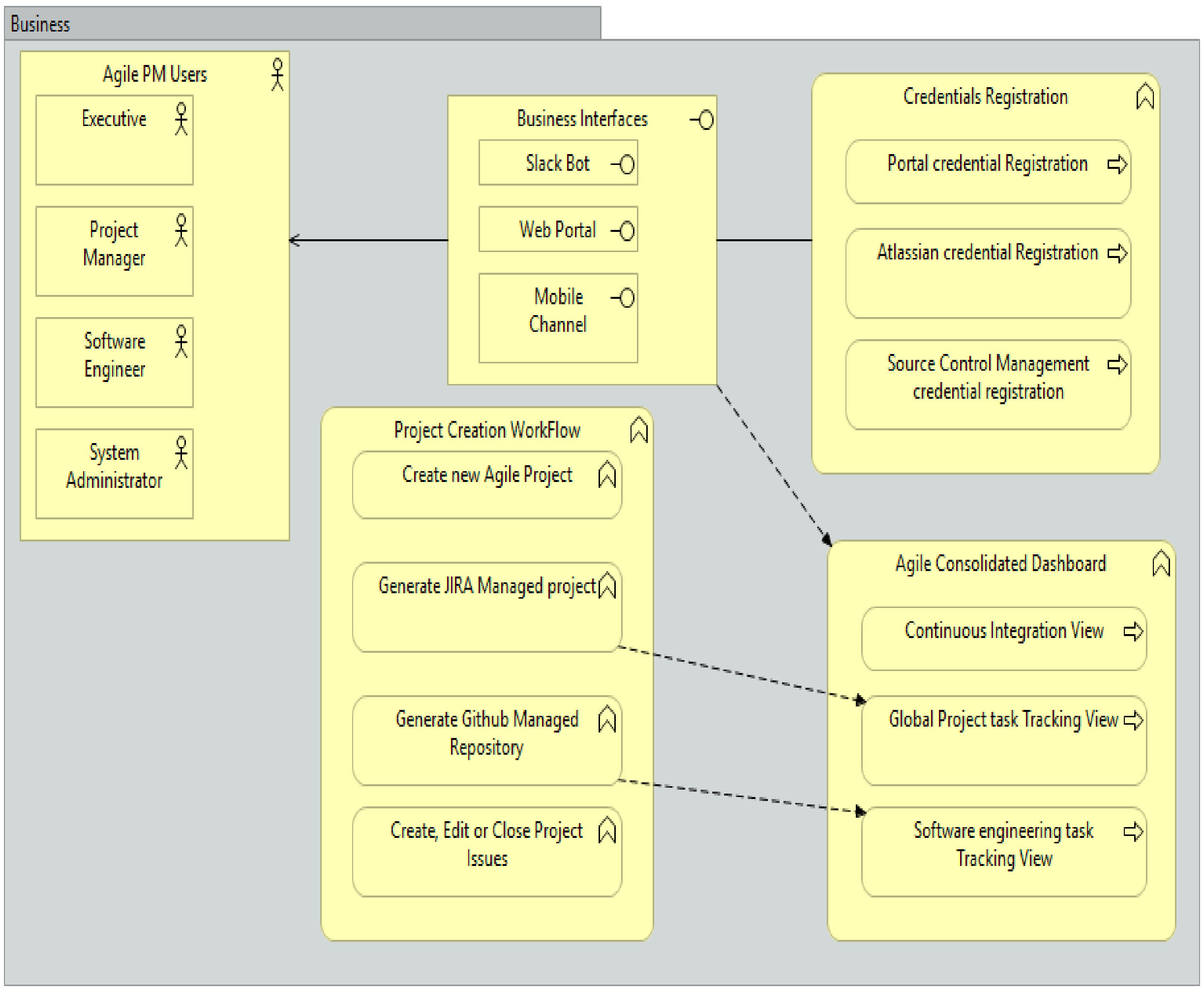

Business
Agile PM Users
Executive
Project Manager
Software Engineer
System Administrator
Business Interfaces
Slack Bot
Web Portal
Mobile Channel
Credentials Registration
Portal credential Registration
Atlassian credential Registration
Source Control Management credential registration
Project Creation WorkFlow
Create new Agile Project
Generate JIRA Managed project
Generate Github Managed Repository
Create, Edit or Close Project Issues
Agile Consolidated Dashboard
Continuous Integration View
Global Project task Tracking View
Software engineering task Tracking View

Figure 5.4: GZ-Agile Project Management (GZAPMC) Consolidator Business View

5.4.2 GZ-Agile Project Management Consolidator: Application View and full picture

The web portal technical design was essentially informed by the Model-View-Controller pattern (*see Appendix B*). Other than the tool implemented, three main backends to which the Agile PM Consolidator would have to integrate were identified: A Source Control Management System, the current Project Management system in use at GZ Consulting Services, and a Continuous Integration and Deployment backend utilised by engineers at the firm.

The system designed comprised of a web portal that also leverages an application programming interface layer (API) which performs some of the integration to third party systems. The front end seamlessly integrates to the source control system via plugins and it leverages similar technologies and an API layer to integrate with the project management system currently used as well as the CI/CD tool.

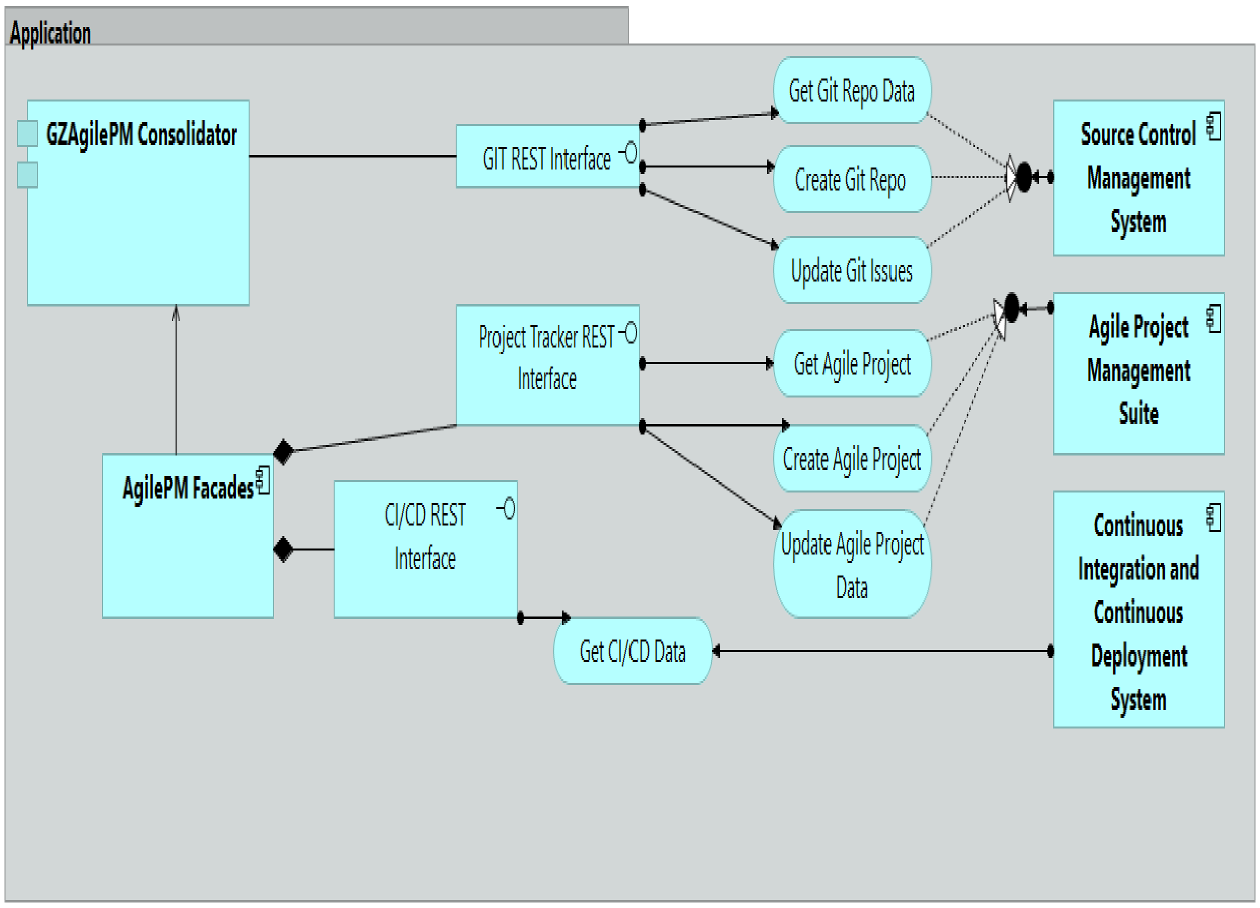


Figure 5.5: GZ-Agile Project Management Consolidator (GZAPMC) Application View

As mentioned in the previous section, figure 5.5 describes how the GZ Agile project management consolidator interacts via APIs calls with the current systems in use at the firm to retrieve, post and aggregate data. The next illustration depicts the overall picture of how the system was designed and includes the technology view considered.

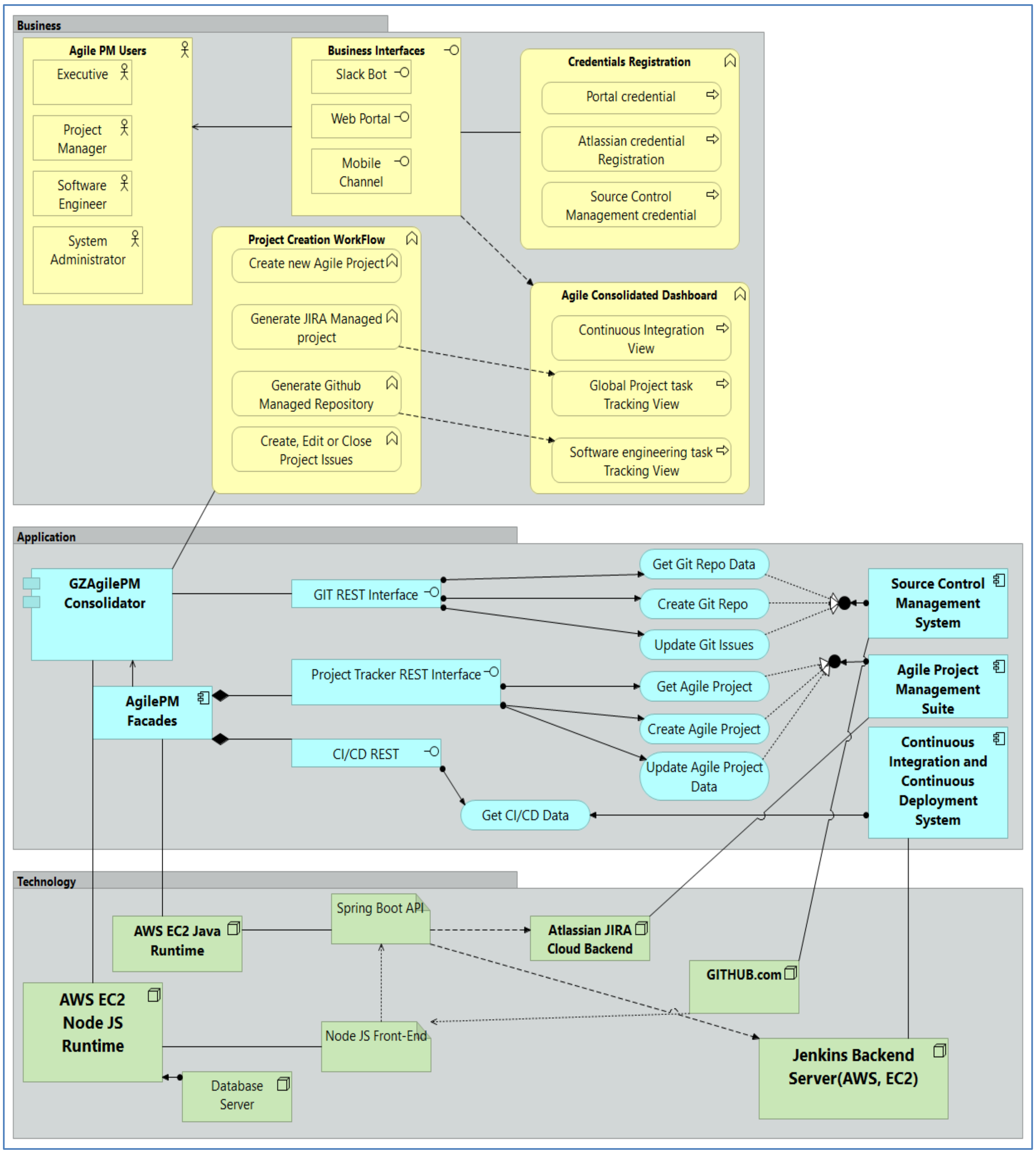

Business
Agile PM Users
Executive
Project Manager
Software Engineer
System Administrator
Business Interfaces
Slack Bot
Web Portal
Mobile Channel
Credentials Registration
Portal credential
Atlassian credential Registration
Source Control Management credential
Project Creation WorkFlow
Create new Agile Project
Generate JIRA Managed project
Generate Github Managed Repository
Create, Edit or Close Project Issues
Agile Consolidated Dashboard
Continuous Integration View
Global Project task Tracking View
Software engineering task Tracking View
Application
GZAgilePM Consolidator
GIT REST Interface
Get Git Repo Data
Create Git Repo
Update Git Issues
Source Control Management System
AgilePM Facades
Project Tracker REST Interface
Get Agile Project
Create Agile Project
Update Agile Project Data
Agile Project Management Suite
CI/CD REST
Get CI/CD Data
Continuous Integration and Continuous Deployment System
Technology
AWS EC2 Java Runtime
Spring Boot API
Atlassian JIRA Cloud Backend
GITHUB.com
AWS EC2 Node JS Runtime
Node JS Front-End
Database Server
Jenkins Backend Server(AWS, EC2)

Figure 5.6: GZ-Agile Project Management Consolidator (GZAPMC) Overall View

Figure 5.6 complements the previous two illustrations of the system by adding the technology stack which forms the foundation of this IS. The entire system runs on Amazon Cloud EC2 Linux infrastructure. Leveraging the selected technologies for the case study prototype, was justified by a need to align with GZCS's existing infrastructure and services stack as described in table 5-2.

The front-end was built with NodeJS to leverage a simple and intuitive MVC paradigm. The internal database is a MYSQL relational database. The front-end also leverages APIS build in the JAVA language. These APIS were designed with the Spring Boot framework and expose collection of façades to call back end services. These facades integrate to GZ Consulting services' Atlassian JIRA cloud back end for project management, GITUHB.com organisational repository to store source and manage engineering tasks, and a Jenkins Backend instance which also resides on Amazon EC2 cloud infrastructure. Figure 5.7 describes a typical user interaction with GZ-Agile Project Management Consolidator via an UML sequence diagram:

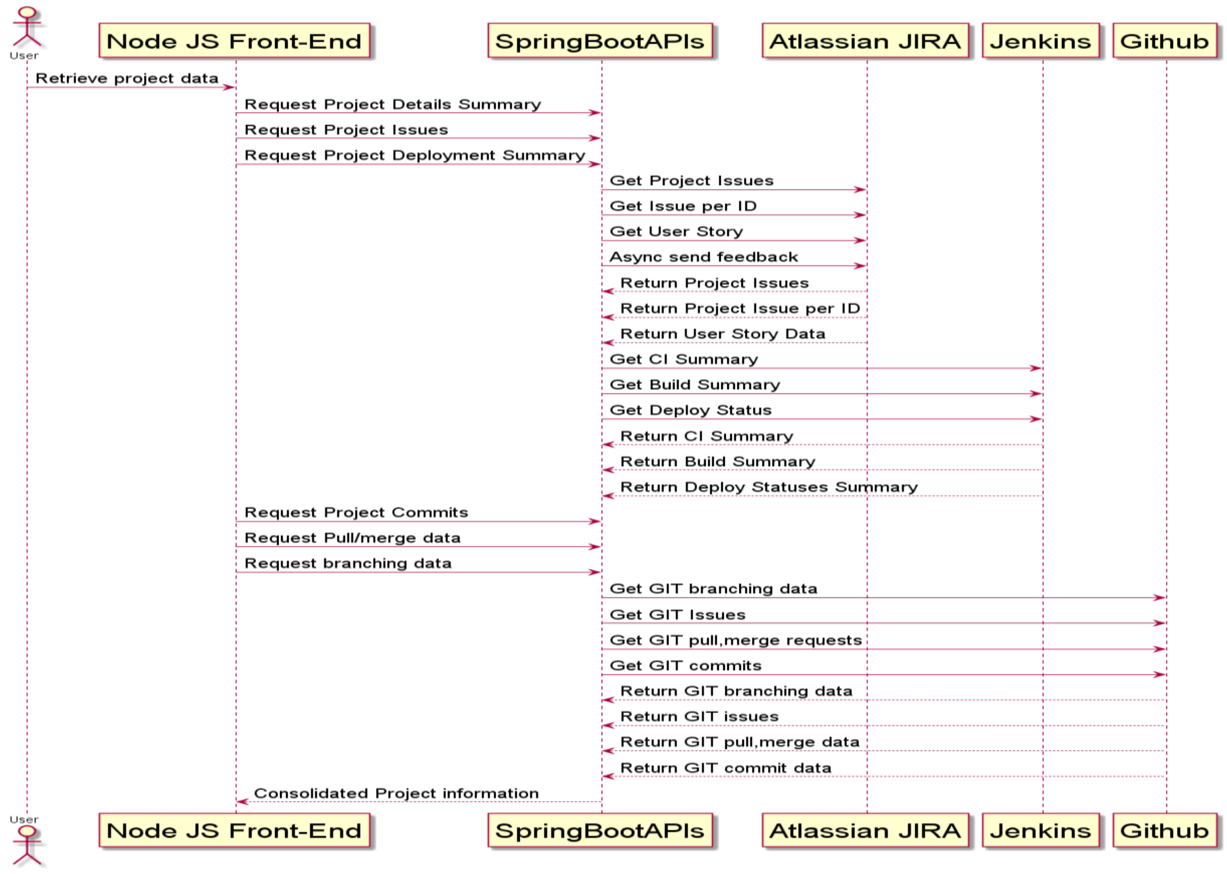


Figure 5.7: GZ-Agile Project Management Consolidator (GZAPMC) Sequence Diagram

5.5 Chapter Five summary

Chapter Five of this thesis attempted to describe the conceptual framework as well as adjustments made to cater for stated research objectives through an activity theory lens. The chapter subsequently proceeded to discuss how technology can be identified as the central mediating tool of an activity system. The researcher then went on to present how a large amount of technology tools in agile oriented organisations can lead to the generation of an increasing amount of information views and how this can pose challenges to project teams.

Chapter Five concluded by further describing the case study at GZ Consulting Services as well as detailing the design of an information systems to address information consolidation from an ArchiMate notation business, application and technology perspective. The IS proposes the integration of cross domain agile engineering activities through a web portal leveraging REST APIS as below:

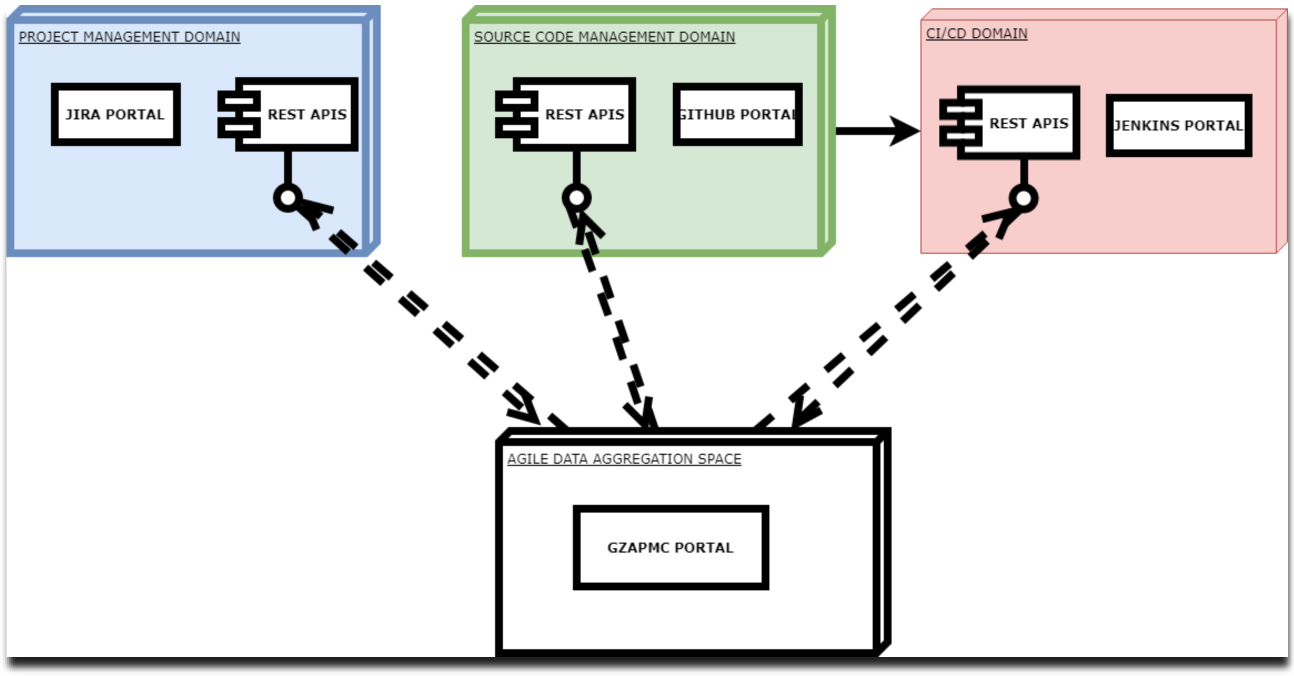


Figure 5.8: Cross domain integration through GZAPMC

In the following parts of this thesis, a discussion on research findings follows. Additionally, the reader might find benefits in referring to appendix C where some of the technology tooling mentioned in the current as well as subsequent chapters is further described.

6 CHAPTER SIX: RESEARCH FINDINGS AND DATA ANALYSIS

Chapter Six discusses findings gathered while conducting the current research exercise. To this end, empirical data retrieved using techniques explored during the research methodology discussion of this thesis is presented.

Data collected through direct observations of GZ Consulting Services staff, semi-structured interviews as well as an electronic survey feedback is discussed, analysed and interpreted. This analysis is intended to contribute to the elaboration of a practical agile oriented framework for the delivery of quality software projects and artefacts.

Considering research sub-questions enunciated in the introductory portion of this thesis, the following themes have so far been addressed through literature review as well as the case study presentation:

- Quality in software settings and the ways in which a modern organisation such as GZ Consulting services deals with this issue.
- The modern technological tools available to agile teams in facilitating quality project deliveries.
- The customising of engineering methodologies selection as well as the importance of tailoring agile processes in modern organisational settings.

While the topics mentioned above were mainly explored through literature review as well as initial interactions with GZ Consulting Services staff, the next section proceeds to a provide a presentation on the research findings.

6.1 Preparing the data analysis

When preparing qualitative data for processing by software analysis, a key requirement for the effective running of this process involves drawing clear enough labels which will serve as a departure point to identify the underlying themes which can be mapped to enunciated research questions (Adu 2016, p. 1).

In qualitative empirical studies, coding is used to arrange data in such a way that a researcher can better gain insight into actionable and relevant information which may not seem apparent at first sight or face value (Smith & Davies in Theron 2015, p. 4). Conceptually, coding describes a natural approach to categorise large amounts of seemingly unstructured content which is often textual in nature (Archer in PsySSA 2018, chap. 2). A code is typically used to capture the essence of a portion of data via some sort of single word or short phrase (Saldaña 2009, p. 3). Consequently, to conduct the qualitative analysis required for this study, four anchor codes were derived from slightly reformulated versions of request questions previously stated as described below:

Table 6-1 Qualitative Analysis Anchor Codes

ANCHOR CODE	**RELATED RESEARCH QUESTION**
Quality software projects expectations	Research question 1: *"What are the expectations in terms of quality project delivery at GZCS and how does the firm try to ensure that quality solutions are delivered to its customers?"*
Agile processes tailoring and agile project challenges	Research question 2: *"Considering the various informal and formal approaches to ensuring the smooth execution of agile processes in software engineering contexts, is GZCS on the right path? How could the firm customize its current processes to better achieve its goals in terms of providing quality software solutions to markets?*"
Technology enhancers for agile projects	Research question 3: *"What are the technological tools your firm uses and/or should be using to ensure delivery of quality projects in a more consistent manner in your opinion?"*
GZAPMC expectations from users	Research question 4: *"GZ Agile PM Consolidator was built to help your organisation have a consolidated view of information in terms of all the tools currently in use at GZCS (JIRA, GITHUB, JENKINS*

	etc.) What are your expectations of this tool?"

As described in table 6-1, interviews were conducted with six staff members to gather their insight. The researcher then applied a themeing coding methodology to perform content and thematic analysis on respondent feedback during this interviewing process driven by the derived anchor codes. The next section proceeds hence to a further discussion on the semi-structured interview findings.

6.2 Semi-Structured interviews findings

Interviewed staff were mainly involved with engineering, management or software project leadership functions. The conversations for these semi-structured interviews were conducted with 2 senior managers, 2 project managers and 2 engineers. During the data collection and analysis process of these interviews, subjects were anonymised and labelled with alphanumeric identifiers; participants were consequently referred to as A1 to A6 during the transcription exercise and the subsequent exploration of data gathered. To further explore results of the researcher's interactions with these participants, the first pieces of insight retrieved are presented in subsequent sections.

6.2.1 Initial findings

Discussions with staff at this stage included interrogations to determine some biographical information on participants, their opinions on agile methodologies and practices at large; as well as the particulars of their applications at GZ Consulting Services.

The first noticeable piece of insight retrieved indicated that almost all interviewed participants had at least 1 to 5 years' experience with agile methodologies and that they enjoyed working in agile settings in most cases. Similar studies have observed a strong correlation between the actual application of agile practices such as Scrum and its positive perception amongst staff members (Salo & Abrahamsson 2008, p. 63). In the present case study, only one staff member indicated that he felt no preference when it came to work in traditional waterfall projects or agile settings.

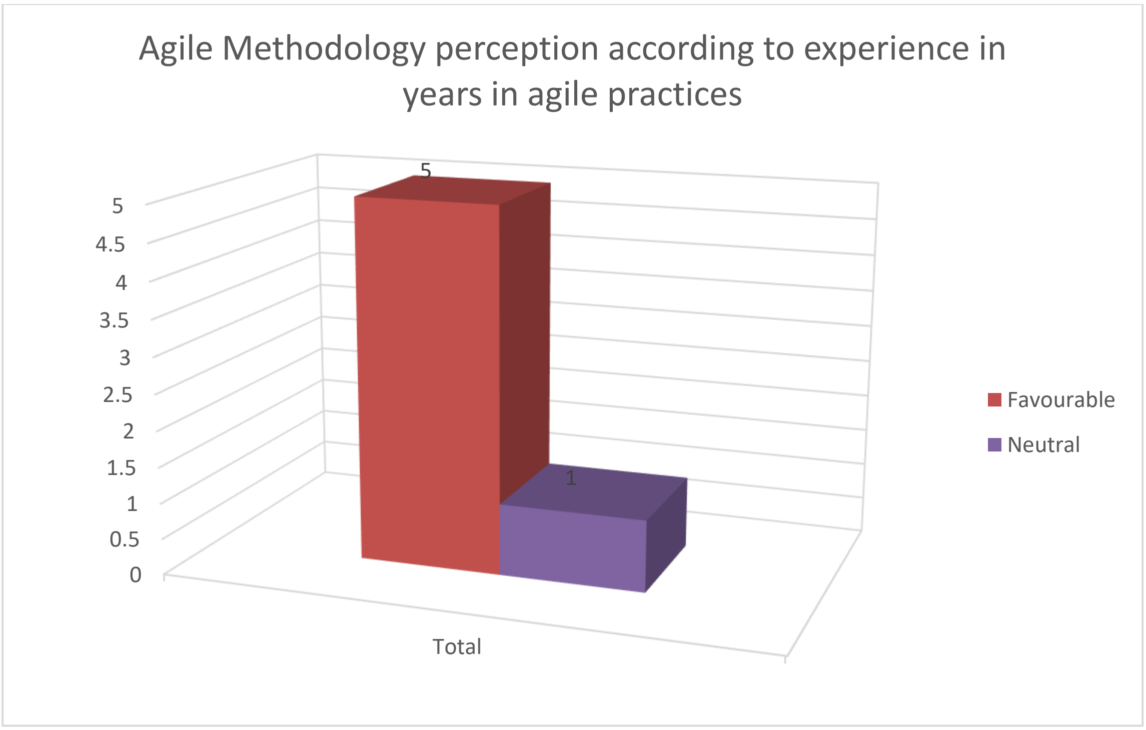


Figure 6.1: Staff appreciation of agile at GZCS

Most participants interviewed had less than 10 years overall software engineering project experience under their belt and did not fit that category. This did not seem to have an incidence on the general positive perception of agile methodologies and the underlying philosophy. To try and further explore their perspectives, the researcher then attempted to gage the satisfaction levels from staff in terms of agile practices implementations at GZ Consulting Services. Considering how employees responded when queried about their opinion in terms of appropriate agile usage at GZCS, figure 6.2 also illustrates how participant's feedback is linked to the number of years experienced in agile settings:

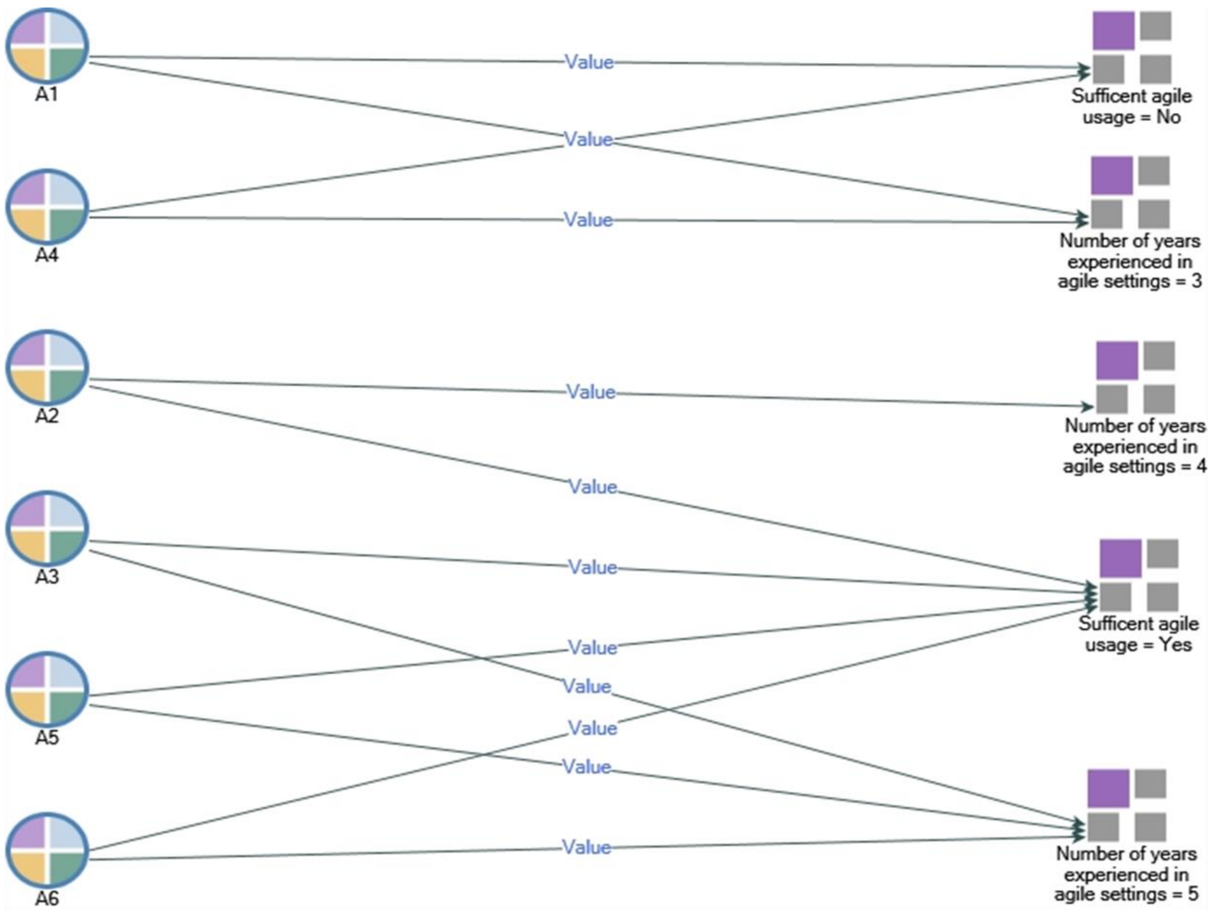

A1
A4
A2
A3
A5
A6
Value
Sufficent agile usage = No
Number of years experienced in agile settings = 3
Number of years experienced in agile settings = 4
Sufficent agile usage = Yes
Number of years experienced in agile settings = 5

Figure 6.2 Agile Practice satisfaction at GZCS

From the generated map above, it emerged that while most interviewed staff evaluated agile practices at GZ Consulting Services as satisfactory or their usage enough; two employees at least were of the opposite view. That both participants (A1 and A4) who believed that the organisation was not leveraging agile in the best possible manner both indicated having three years' worth of experience in agile environments. The rest of the interviews all had four or more years in agile settings. An interesting piece of insight observed was that participants A1 and A4 were the two engineers interviewed. It emerged that these participants were essentially referring to technology usage and engineering practices rather than agile philosophy implementation per se. They for instances indicated that they believed there was not enough automated testing at the firm or efforts to promote test-driven approaches such as TDD in projects. When quizzed further by the researcher in an attempt to clarify and expand on their responses, the subjects explained that they believed the organisation was actually doing a good job in terms of applying agile practices and tooling in their view, but some progress was still necessary for the engineering functions specifically in terms of testing practices, it hence appeared that ultimately interviewees were mostly satisfied with the agile culture and practices at GZCS.

Considering the research framework for this thesis as discussed in previous chapters as well as the anchor codes established, inductive coding and data themeing were leveraged to explore feedback from participants. In the next section of Chapter Six, this process is described in more detail.

6.2.2 Inductive coding and subsequent themeing

For this thesis's interview transcripts data analysis, the researcher first leveraged a deductive strategy. This technique was used to derive anchor codes from research questions enunciated in Chapter One. Researchers typically adopt such an approach to formulate a coding scheme. The coding scheme in question will drive subsequent qualitative data analysis processes before most data is actually collected in a top-down manner (Nixon 2014).

Next, after the data was collected, an inductive approach was used to code participant transcribed feedback verbatim or in vivo as it is often described in qualitative analysis contexts (Manning 2017, p. 2). This was done in a bottom up manner as inductive coding typically requires that identified code be driven from the data structure and content itself (Beckwitt 2016, para. 3).

Finally, themeing was leveraged to continuously explore and systematically refine participants' feedback interpretation. This was done by establishing thematic identifiers to better understand the underlying patterns of information not easily observed during a superficial observation of data collected from staff during semi-structured interviews and other informal exchanges between participants and the researcher.

Themeing data consists in identifying and labelling significant information (such as the anchor codes generated earlier), looking at its possible meaning or interpretations, and then deriving shorter phrases to further describe it (Adu 2018). It additionally helps researchers ground data gathered into their interpretative framework and draw meaningful insight during the analysis process (Saldana in Martin 2015, p. 9).

To further achieve a better understanding, some qualitative methodology domain experts have suggested that short generic coding is not sufficient; their recommendation is to analyse and label the data with extended thematic statements (Saldaña 2009, p. 139). Ergo, applying themeing analysis to the initial anchor codes identified in table 6-1 and the data gathered, the following additional sub-categories were derived:

- **Quality software project expectations**
    - Quality software project delivery implies good communication across teams and between all stakeholders.
    - Quality software project delivery implies efficient task assignment.
    - Quality software project delivery implies alignment to customer needs and understanding their requirements and their organisation.
    - Quality software project delivery implies a focus towards excellence.
    - Quality software project delivery implies timeliness.
    - Quality software project delivery implies customer satisfaction and collaboration.
- **Agile processes tailoring and agile project challenges**
    - Tailoring agile processes suggests a full implementation of agile methodologies.
    - Smaller agile companies such as GZCS have difficulties with tight resources.
    - Smaller agile companies such as GZCS need to optimize their processes to their resources.
    - Organisations should use formal project management and not use agile processes as an excuse for anarchy.
- **Technology enhancers for agile projects**
    - Agile productivity tools including the Atlassian suite with JIRA and Confluence, Slack, Jenkins, GITHUB as well as the proposed prototype post implementation at GZ Consulting Services: *GZAPMC*.
- **GZAPMC expectations from users**
    - Tackle the issue of multiple interfaces to project data.
    - Quality management enhanced functionality.
    - Integrated direct communication channel and improved intuitive task assignment functionalities.
    - Integration to a bigger array of external tooling.

In the next section, a content analysis discussion of these themes and derived sub-categories follows.

6.2.3 Findings discussion

To improve visualization of textual data which requires processing for analysis in qualitative contexts, word clouds have been found to be very useful tools for researchers (Heimerl *et al.* 2014, p. 1833). In qualitative studies which require data analysis of textual data, such instruments are used to depict intuitively a summarized illustration of emerging important terms (Wei *et al.* 2015, p. 539). They rely on word count frequency analysis which typically results in a display of key words where font size is supposed to be proportional to how often the word is used (Turner 2017, para. 1).

These visual tools are basically graphic illustrations of key word frequency which can be customized based on some excluding or including criterion determined by researchers (Vasconcellos-Silva, Carvalho & Lucena 2013, para. 3).

To proceed with analysis of data gathered in semi-structured interview conducted during the current study, transcripts were cleaned for processing to generate the word cloud below:

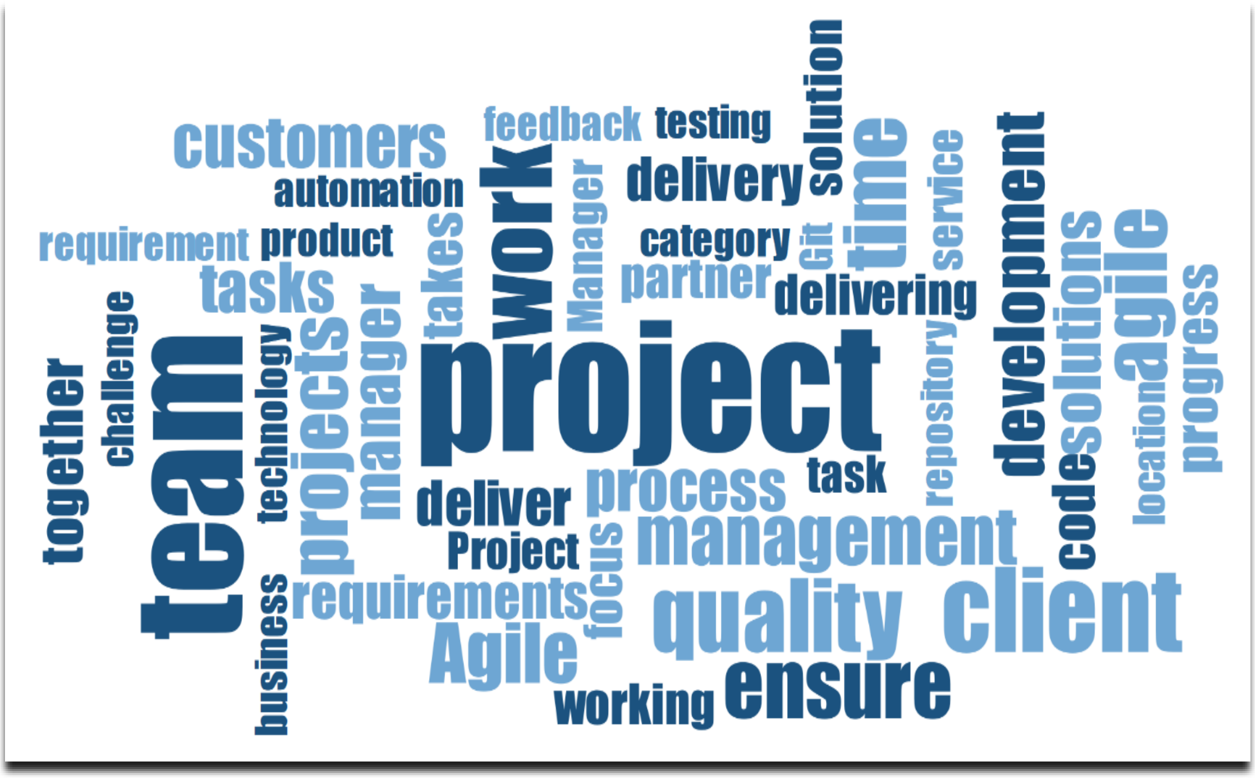


Figure 6.3 Interview transcript word cloud

Some of the key terms which seemed to emerge from interview transcripts included engineering and software project management jargon such as *"project"," work"," team", "delivery," delivery", "challenge", "quality", "client"* etc.

These terms were used by both management and engineering staff alike and were an integral part of the organisational day to day lexicon at large but more particularly in the context of engineering projects delivery. Table 6-2 illustrates a frequency distribution for the top six words with a length equal or superior to four characters emerging from this word cloud:

Table 6-2 Top 6 terms from interview transcripts

Word	Word length	Frequency	%	Rank
Project	7	36	8,61	1
Team	4	31	7,42	2
Client	6	21	5,02	3
Work	4	18	4,31	4
Quality	7	17	4,07	5
Agile	5	15	3,59	6

The first three items in this list were encountered 88 times in the transcript and account for around 64% of the most used terms. Consequently, the next sections proceed to discussion on these terms' usage in context of the conducted semi-structured interviews.

The most recurrent words used after frequency analysis of interview transcripts were "*Project"* and *"Team"*. Table 6-3 and 6-4 illustrate examples in which it was used when participants were interrogated about their work environment and expectations.

Table 6-3 "Project" usage instances

PROJECT
(…) amongst developers and **PROJECT** managers (...)
(…) enough to ensure quality **PROJECT** delivery. All the above-mentioned (…)
(…) Project Manager I am a **PROJECT** coordinator/administrator (…)
(…) Manager I am the main **PROJECT** owner as far as the IT team (…)
(…) from the beginning of the **PROJECT** all the way to final delivery (…)
(…) always with an accounts **PROJECT** manager (…)
(…) to understand progress on **PROJECT** and feeds this back to the client (…)
(…) at some point for a **PROJECT,** the marketing department (…)
(…) all items to tick off in the **PROJECT** through excel (…)
(…) then takes it to **PROJECT** delivery (…)
(…) Whatever best **PROJECT** management (…)
(…) You expect your **PROJECT** to be delivered on time and quality (…)

Table 6-4 "Team" usage instances

TEAM
(…) hand with customers, making our **TEAM** available for support (…)
(…) the backend **TEAM.** Tools we should be using is one that (…)
(…) as far as the IT **TEAM** is concerned (…)
(…) pushing perfection on the **TEAM**, we do however strive for (…)
(…) strive for excellence. Our **TEAM** is split into front end (…)
(…) as our tech **TEAM** (…)
(…) **TEAM** together (…)

As illustrated above, interviewees indicated that they always worked within the context of a collaborative enterprise. This was done in terms of projects and under the leadership of project managers, team leads, project coordinators well versed in traditional project management practices and more agile methodologies as well. The collaborative aspect of their work is best illustrated by the emphasis noticed on usage of the term “team”. This seemed to perhaps indicate a grasp of the need for increased coordination and cooperation in the objective of successfully running and delivering their projects.

When quizzed about the way the organisation attempted to meet customer needs, the next two most used words by staff included the terms *"Client"* and *"Work"* as described in tables 6-5 and 6-6:

Table 6-5 "Client" usage instances

CLIENT
(…) systems work for **CLIENT** betterment. Our systems are built for custom (…)
(…) delivering the underlying **CLIENT** challenge, and that message being (…)
(…) encourages **CLIENT** interaction on a front-end (…)
(…) **CLIENT** facing and directly affect (…)
(…) quality solutions to the **CLIENT** (…)
(…) first meeting with the **CLIENT** (…)
(…) a solution is drawn up for the **CLIENT** (…)
(…) back to the **CLIENT** to present and receive feedback (…)
(…) **CLIENT** is happy with the solution (…)
(…) feeds this back to the **CLIENT.** The Account manager also ensures (…)

Table 6-6 "Work" usage instances

WORK
(…) I prefer agile because my **WORK** is much more focused (…)
(…) system to improve **WORK** productivity (…)
(…) proud to say we **WORK** to align ourselves with customer (…)
(…) upgrading systems **WORK** for client betterment (…)
(…) quality **WORK** is key for our survival (…)
(…) these two ends must **WORK** together (we strive for that) (…)
(…) front end and the back-end **WORK** closely together (…)
(…) delivered quality **WORK** that everyone was happy with (…)

Usage of these terms mainly seemed to describe project related activities executed by staff as well as their outputs in certain instances. What also seemed to emerge with the recurrence of the word "*client*" was clear focus on customer needs and a client centric approach to delivering solutions.

The final two most used words derived from the frequency list previously described included *"Quality"* and *"Agile"* as described below:

Table 6-7 "Quality" usage instances

QUALITY
(…) long meetings. In terms of **QUALITY** software projects delivery (…)
(…) developing and providing **QUALITY** software (…)
(…) excellent enough to ensure **QUALITY** project delivery (…)
(…) We try to ensure **QUALITY** solutions by working hand in hand (…)
(…) all that we do, and **QUALITY** work is key for our survival (…)
(…) we guarantee our clients **QUALITY** output (…)
(…) strive for that to ensure **QUALITY** solutions are delivered (…)
(…) deliver **QUALITY** solutions to the client (…)
(…) manager also ensures **QUALITY** control (…)
(…) organization to ensure **QUALITY** standards are constantly met (…)

Table 6-8 "Agile" usage instances

AGILE
(…) I prefer **AGILE** because my work is much more focused (…)
(…) I have mostly worked in **AGILE** organisations though I don't
(…) we are a small firm thus **AGILE,** able to deliver at a speed (…)
(…) always worked in **AGILE** settings (…)
(…) working code through the **AGILE** model (…)
(…) adds a layer for the **AGILE** project's management (…)
(…) I am an **AGILE** and scaled agile advocate (…)
(…) We have adopted AGILE approach where we incrementally develop (…)
(…) our **AGILE** process has been further optimized (…)

As mentioned during literature review (section 2.6.4), informed by chain theory and aspects of Aristotelian metaphysics discussed in Chapter Five (section 5.4.1), one could argue that improving the process by which software projects are executed and managed could improve the delivery of quality solutions. Throughout the current thesis, this suggested to be possible through the adoption and implementation of agile methodologies, practices and technological tools. Consequently, additional terms which were frequently encountered during conversations with GZ Consulting Services staff could be explored to further discuss this possibility. For example, featured in the top twenty most used terms during analysis of interview transcripts were words including "*Management", "Process" and "Deliver"*. Tables 6-9, 6-10 and 6-11 proceed to give a couple of instances where these terms were used.

Table 6-9 "Management" usage instances

MANAGEMENT
(…) input suggestions. Project **MANAGEMENT** tools with timeline (…)
(…) projections. Quality **MANAGEMENT** – a tool with longevity (…)
(…) takes this up with **MANAGEMENT** in the organization to ensure (…)
(…) someone in a senior **MANAGEMENT** role, who takes (…)
(…) unavailable thus senior **MANAGEMENT** must step in (…)
(…) Whatever best project **MANAGEMENT,** real time client service (…)
(…) formal project **MANAGEMENT** approach because (…)
(…) specific project **MANAGEMENT** methodology. I am forced (…)
(…) approach to project **MANAGEMENT** involves implanting process (…)
(…) for the agile projects **MANAGEMENT** that interacts (…)
(…) issue tracking and project **MANAGEMENT.** Git: The project team (…)

Table 6-10 "Process" usage instances

PROCESS
(…) feedback for the development **PROCESS,** we find that it takes too long (…)
(…) involves implementing **PROCESSES** and controls to ensure (…)
(…) we do have a formal **PROCESS** used to execute formally (…)
(…) tight. As a result, our agile **PROCESS** has been further optimized (…)
(…) testing, software building **PROCESS,** software deployment process (…)
(…) ensure that the creation **PROCESS** executed faster (…)

Table 6-11 "Deliver" usage instances

DELIVER
(…) assigned their tasks and **DELIVER** them on time (…)
(…) we **DELIVER** strategic solutions (…)
(…) able to **DELIVER** quality solutions to the client (…)
(…) able to **DELIVER** at a speed (…)

(…) achieve all of these and **DELIVER** on all the projects (…)
(…) you know they will be able to **DELIVER** a quality software project (…)
(…) implemented in order to **DELIVER** our product much faster (…)

The term "*Management*" was predominantly used to describe the handling of software engineering projects at the firm by project leadership. The term was used in this manner by both project leadership and software engineering staff members. In one instance it referred to an exchange with the researcher about how the firm could better deal with delivering quality solutions by leveraging practices from Quality Assurance and Quality Management. Most often, the term was used in the context of software projects and project management as a practice at large.

"*Process*" was mostly used to describe the activities and actions executed by staff in the context of producing solutions during projects while "*Deliver*" was the main verb employed by participants in expressing their emphasis on providing quality software artefacts in a timely manner to their customers. Before moving on to present observational insights gathered during data collection, the next section proceeds to a brief synthesis of information retrieved during semi-structured interviews at this stage.

6.2.4 Semi-structured interviews summary

The first step of collecting data for this research involved conducting interviews in a semi-formal manner with selected GZ Consulting Services staff. Both engineering, management and project leadership functions were involved in these exchanges with the researcher. After biographical information was retrieved, the interviewer sat with participants at their workplace and went about trying to establish if a bias could be perceived towards agile practices in genera. For example, after determining a subject's level of experience in software projects, one of the questions asked was formulated simply formulated as "*Do you prefer working in agile or traditional SDLC settings? Can you please explain your response?*". To understand how participants perceived the notion of quality in their organization, another question which was put to them for instance, was formulated similarly to the following: *"In terms of delivering products to your customers, what are your expectations as stakeholder? How does GZCS ensure their customer base is satisfied?"*. These were essentially the kind of open-ended questions put to participants. The next section proceeds to discuss insights emerging from these exchanges.

Key recurring terms emerging from interview transcripts revolved around *project, team, client, work, quality* and *agile.*

The most apparent piece of insight seemed to show a direct correlation between the number of declared years experienced in agile environments by participants and their enthusiasm or apprehension with agile practices. Most employees interviewed seemed to view agile in a positive light and found the usage of agile practices effective at the firm. The analysis of interview transcripts seemed to show that participants viewed good communication and effective team collaboration with a client centric approach as the most appropriate manner to ensure the delivery of quality solutions in context of agile projects. To further gain insight into information retrieved during the data collection, findings emerging from observing day to day activities at GZ Consulting Services are presented next.

6.3 Observing agile processes

Underpinned by an Activity Theory centred perspective, Chapter Five's proposed conceptual framework identified the key activities to be observed:

- Engineering related activities (Software/DevOps)
- Agile projects management related activities (Stand-up meetings, ChatOps communications, Kanban board management etc.).

These activities are essentially encompassed by project processes in an engineering setting. They are additionally subject to adaptations for specific needs and are often customized and customized to GZCS's evolving organisational needs.

During interactions with GZ Consulting Services senior staff, it appeared that this was a view understood and adopted by management. Some Project leaders explained for instance that although their principal project management tool was Atlassian's JIRA suite, they mostly only leveraged Kanban functionality and not necessarily other agile features such as Epics or Themes. In following sections of Chapter Six, a discussion on observations of staff during day to day activities follows. These observations conducted concern agile processes executed at GZ Consulting Services prior and post the IS prototype's implementation during the case study conducted for this thesis.

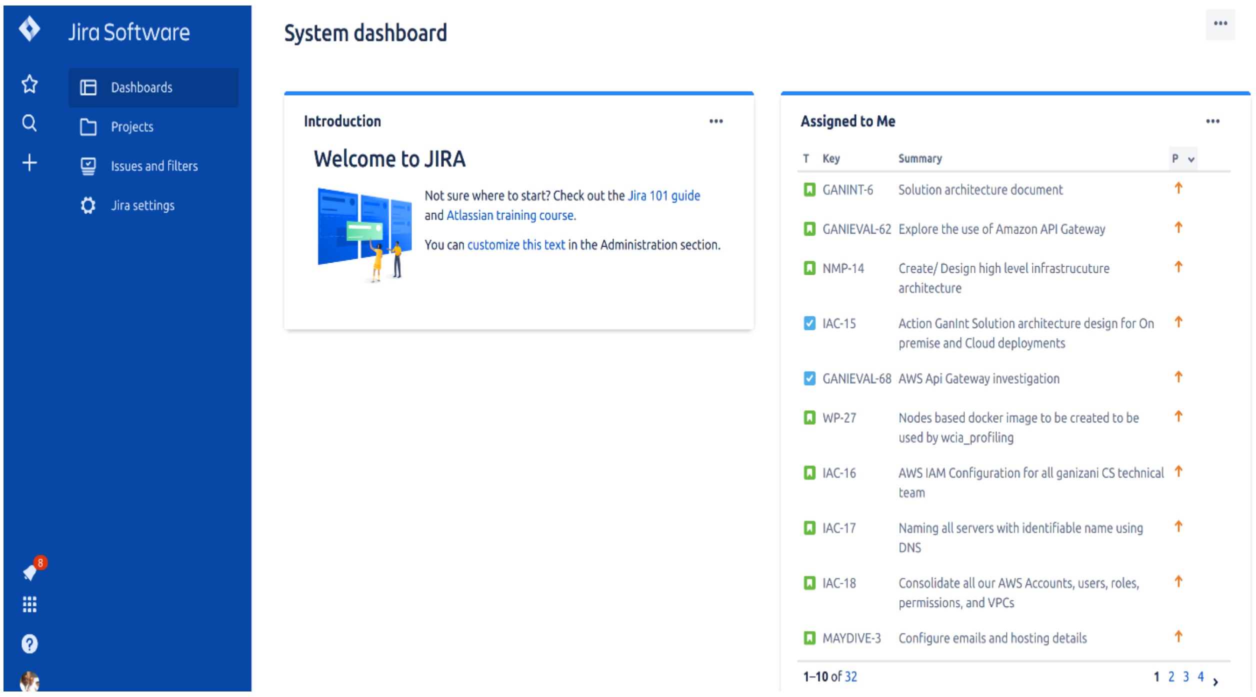
Subsequent portions of the research observational process hence illustrate a use case where project team members interact with the organisation's central project management tool:

Atlassian’s JIRA; the source code management system: GITHUB.com, the ChatOps application: Slack and the central automation build and deployment system: Jenkins.

The following sequence of described events occurred between August and September 2017.

6.3.1 GZCS Software Project inception pre-implementation: Leadership perspective

During the current specific use case, observations of a GZ Consulting Services project leader tasked with managing a new business initiative is conducted. The first staff member observed is identified as “***PM1***” and is responsible for driving a new time management improvement process supported by the JIRA corporate web portal. After strategic meetings with relevant executives and operational staff, **PM1** proceeds to login to the corporate JIRA portal to formalize the new time management system initiative.



System dashboard

Introduction

Welcome to JIRA

Not sure where to start? Check out the Jira 101 guide and Atlassian training course.

You can customize this text in the Administration section.

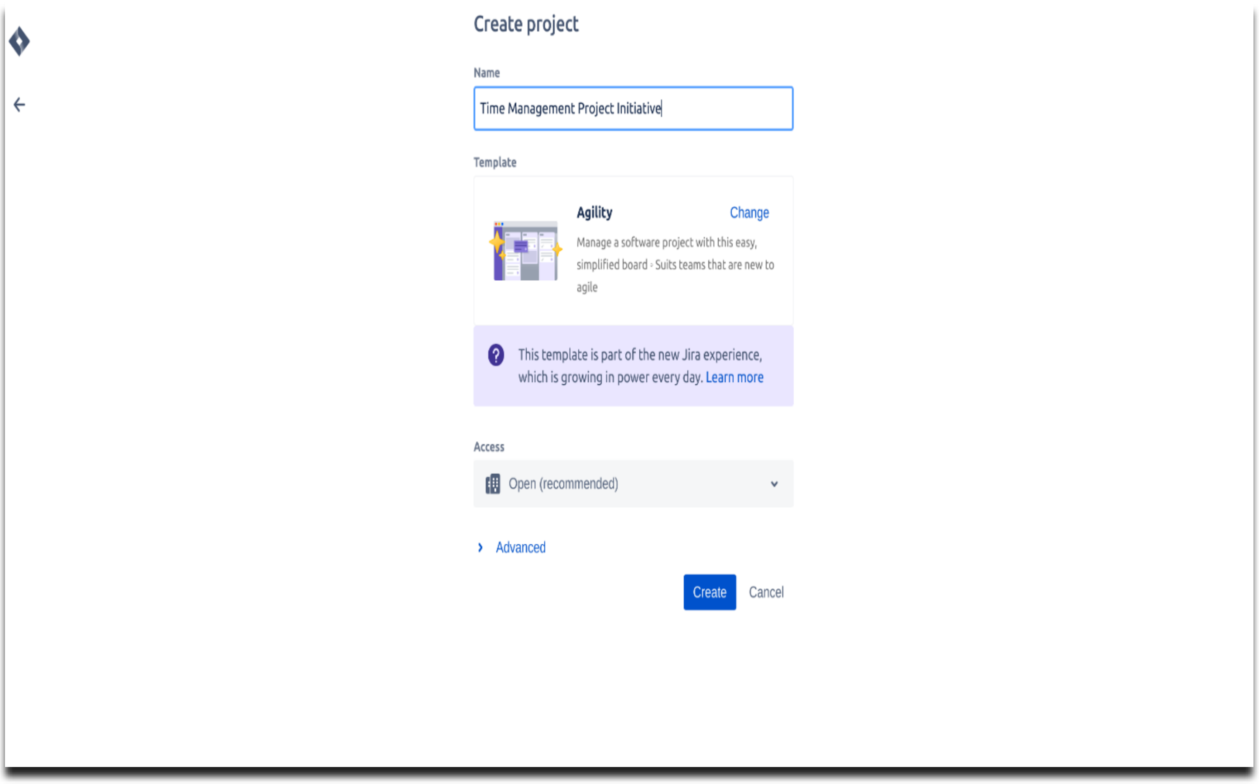
Assigned to Me

T	Key	Summary	P
☐	GANINT-6	Solution architecture document	↑
☐	GANIEVAL-62	Explore the use of Amazon API Gateway	↑
☐	NMP-14	Create/ Design high level infrastrucuture architecture	↑
✓	IAC-15	Action Ganlnt Solution architecture design for On premise and Cloud deployments	↑
✓	GANIEVAL-68	AWS Api Gateway investigation	↑
☐	WP-27	Nodes based docker image to be created to be used by wcia_profiling	↑
☐	IAC-16	AWS IAM Configuration for all ganizani CS technical team	↑
☐	IAC-17	Naming all servers with identifiable name using DNS	↑
☐	IAC-18	Consolidate all our AWS Accounts, users, roles, permissions, and VPCs	↑
☐	MAYDIVE-3	Configure emails and hosting details	↑

1–10 of 32 1 2 3 4 ›

Figure 6.4 GZCS JIRA Home page

From the home page, **PM1** went on to create a JIRA Project artefact for the new business initiative in question:


Create project

Name

Time Management Project Initiative

Template

Agility — Change

Manage a software project with this easy, simplified board › Suits teams that are new to agile

This template is part of the new Jira experience, which is growing in power every day. Learn more

Access

Open (recommended)

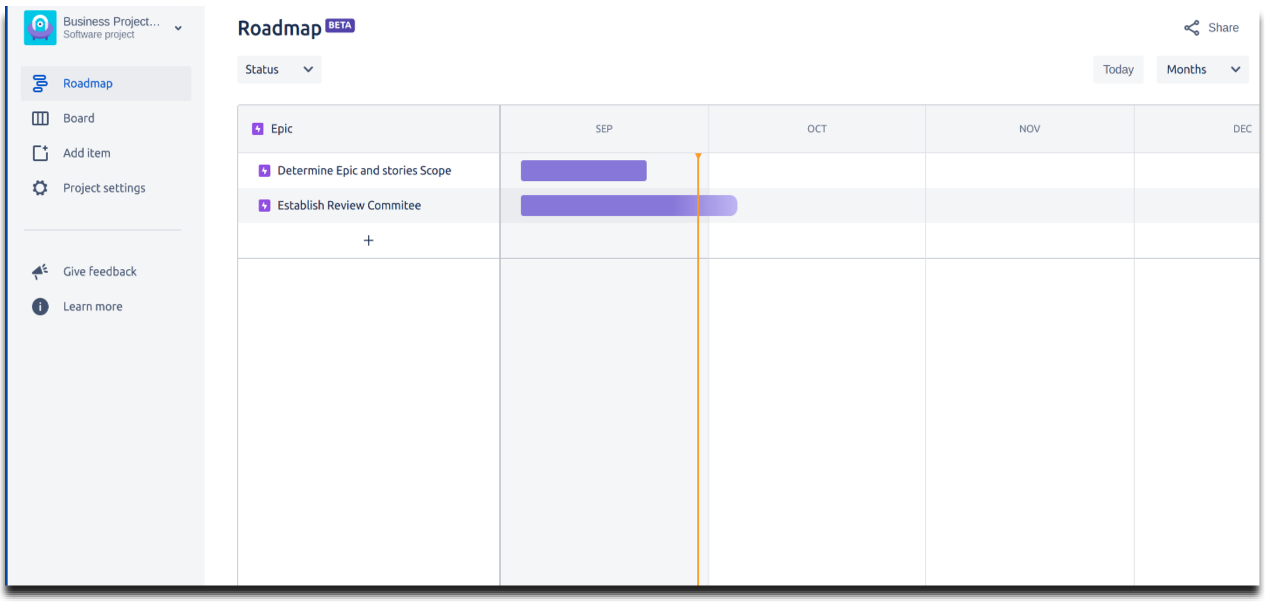
Advanced

Create Cancel

Figure 6.5 Jira Project Creation

After creation of the initial project artefact, JIRA generated two sub-items: a generic strategic roadmap as well as an overall Kanban like board to improve visualisation of the activities to be completed for this new initiative.

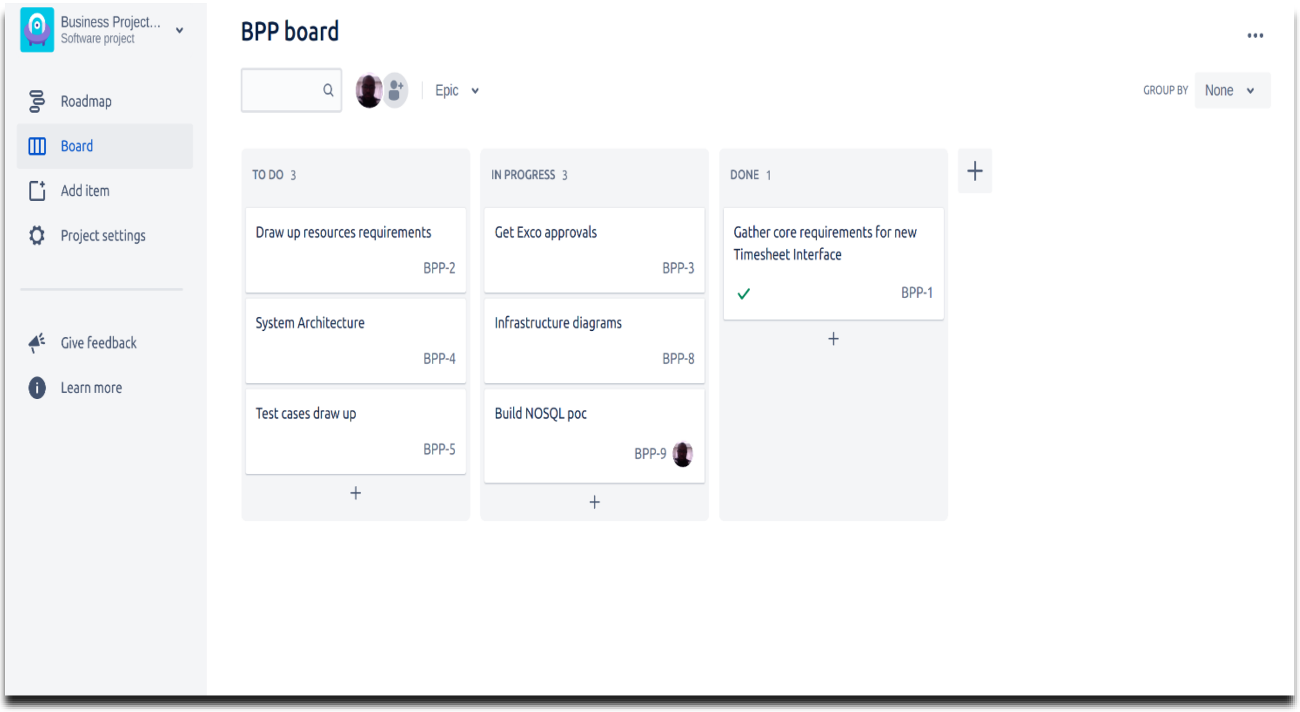
On the road map artefact, **PM1** created an agile epic with the first key items deemed necessary for project initiation including activities such as determining epic and user stories scopes; then on the JIRA default Kanban board, more specific items necessary to run agile projects such as resources scoping, infrastructure design, prototype development were identified. These first activities are illustrated as below:



Determine Epic and stories Scope

Establish Review Commitee

Figure 6.6 Jira Project Road Map



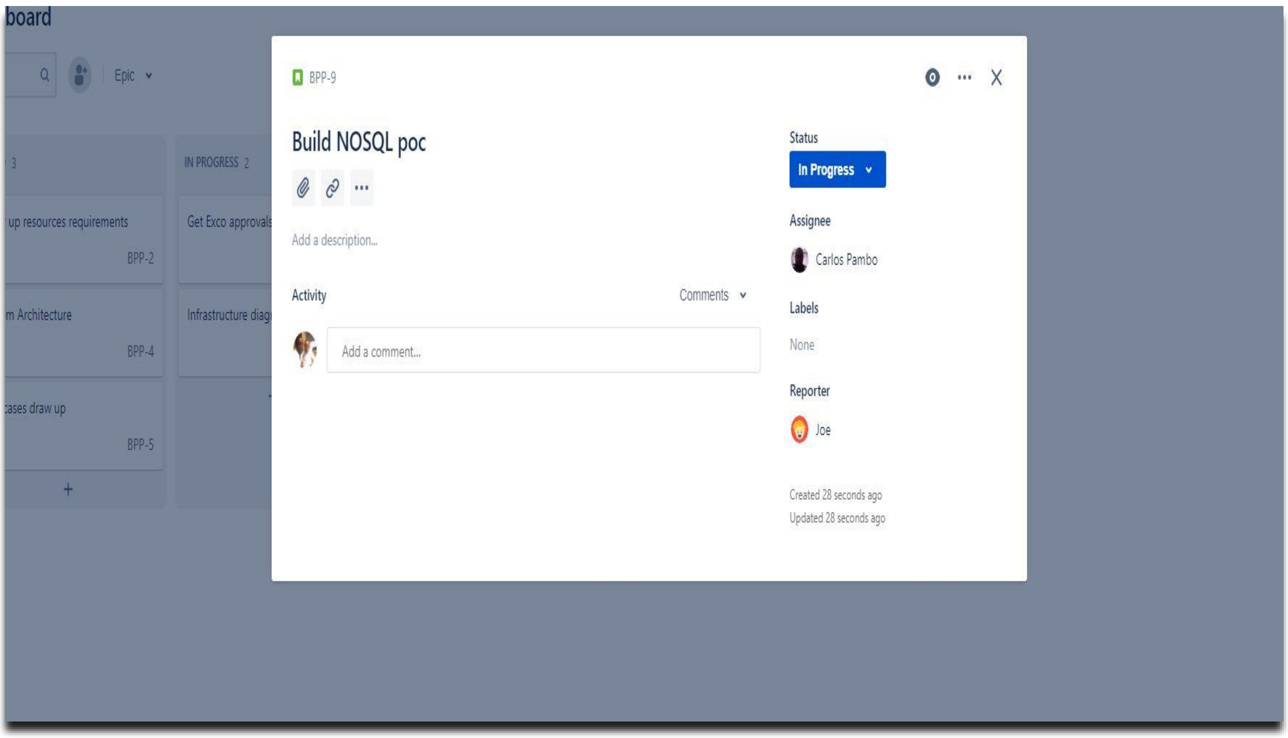
TO DO 3
- Draw up resources requirements BPP-2
- System Architecture BPP-4
- Test cases draw up BPP-5

IN PROGRESS 3
- Get Exco approvals BPP-3
- Infrastructure diagrams BPP-8
- Build NOSQL poc BPP-9

DONE 1
- Gather core requirements for new Timesheet Interface ✓ BPP-1

Figure 6.7 GZCS Project Kanban Board Artefact

PM1 was able to from this point on, to drill down into specific activities on the Kanban board, assign them to project team members and enquire on progress during stand-up meetings and scrum sessions. Task statuses can be updated by default to “In Progress” or “Done” after having been created under the “TO-DO” label.


board

Epic

3

IN PROGRESS 2

up resources requirements BPP-2

Get Exco approvals

m Architecture BPP-4

Infrastructure diag

cases draw up BPP-5

BPP-9

Build NOSQL poc

Add a description...

Activity

Comments

Add a comment...

Status: In Progress

Assignee: Carlos Pambo

Labels: None

Reporter: Joe

Created 28 seconds ago

Updated 28 seconds ago

Figure 6.8 Creation and assignment of JIRA activity

Typically, after a task is issued and assigned to a project team member, outside of stand-up meetings and scrum sessions, the engineering workflow is completely transparent to **PM1**. The project resource to whom this task has been allocated in this scenario will typically receive some sort of notification via his or her corporate email account or ChatOps interface.

In the next section hence, Chapter Six proceeds to describe how the process was handled from an engineer's point once a task request has been received.

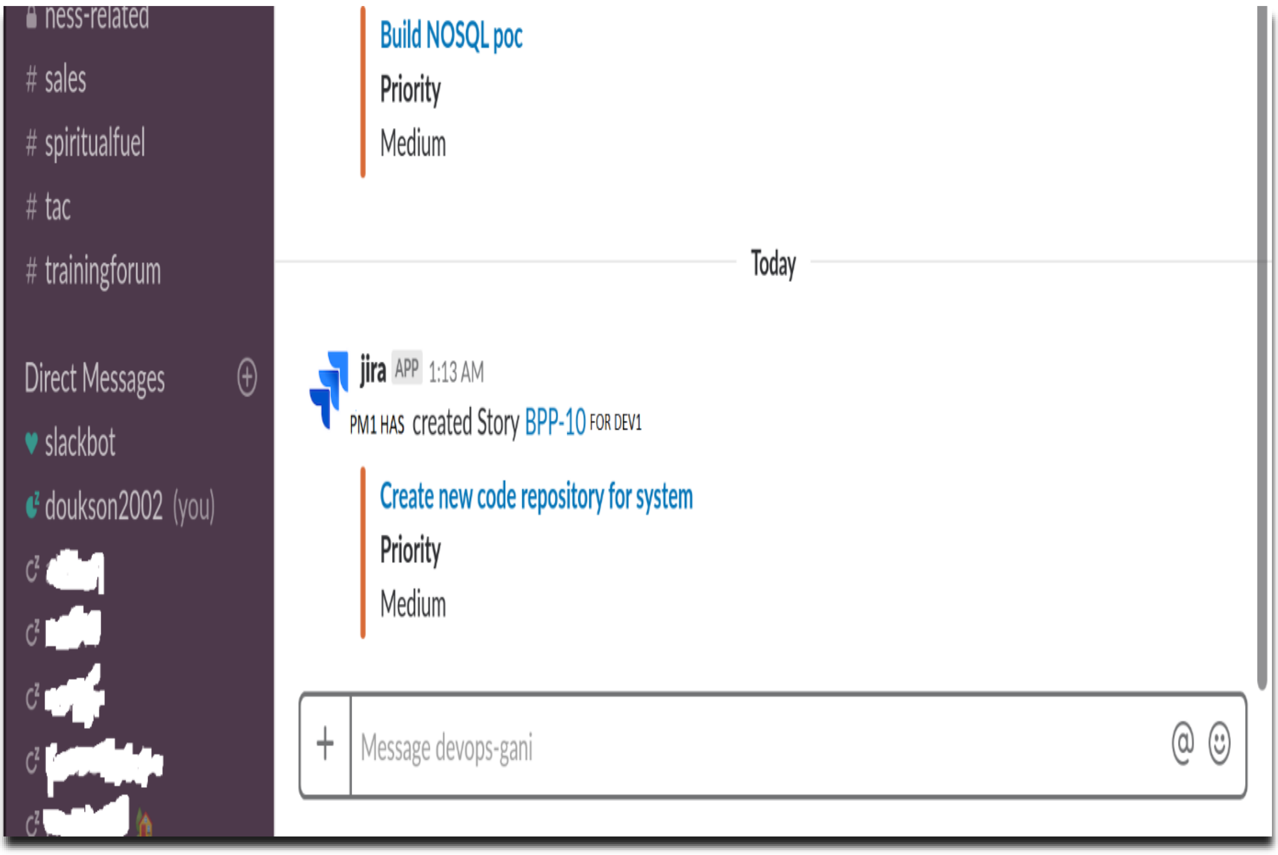
6.3.2 GZCS Software Project execution pre-implementation: Engineering viewpoint

As discussed during the third chapter of this thesis (section 3.1.5), in enhanced agile environments, engineering teams engaged in reducing the expectational gaps between engineering and operations opt for the adoption of DevOps and related the technological tools. Consequently, organisations are increasingly relying on ChatOps.

ChatOps tools are very attractive to agile teams because they enhance DevOps by improving communication between engineering staff and project leadership. They help organisations achieve this by increasing visibility for each project. This tooling enables all team members see activities actioned by colleagues and hence provides a clear view of what is occurring across environments through a concise and direct channel (Juopperi 2017, p. 3).

At GZCS, the main communication mechanism teams engage in for ChatOps is an application recently acquired by Atlassian called Slack. The system has been configured to send notifications when tickets are created on JIRA.

Following the scenario described in section 6.3.1, the use case discussion follows from the point where an engineering team member identified as “**DEV1**” received a notification from the JIRA project management portal via a customized Slack instant messaging channel.


ness-related
# sales
# spiritualfuel
# tac
# trainingforum

Direct Messages
slackbot
doukson2002 (you)
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]

Build NOSQL poc
Priority
Medium

Today

jira APP 1:13 AM
PM1 HAS created Story BPP-10 FOR DEV1

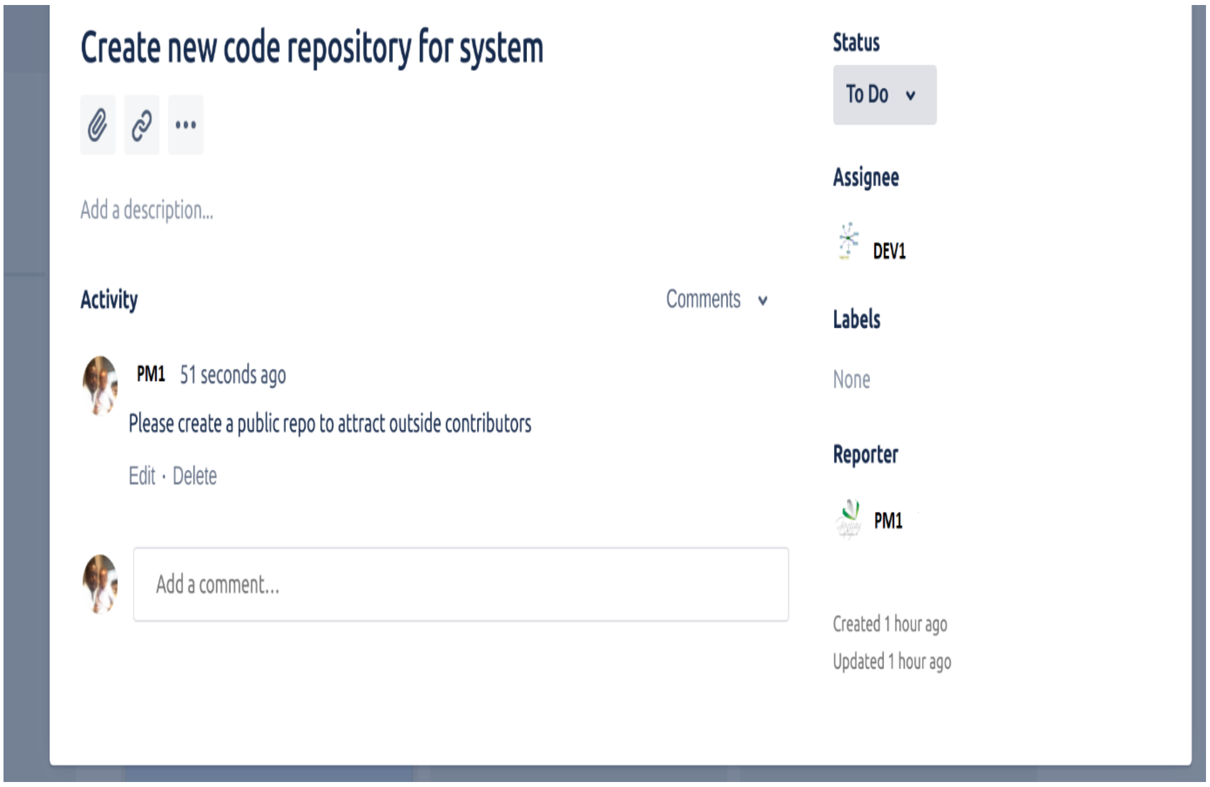
Create new code repository for system
Priority
Medium

Message devops-gani

Figure 6.9 Reception of assigned task via Slack channel

After receiving notifications as depicted above via the Slack interface channel, **DEV1** was required to navigate to his corporate JIRA board. This is needed to ascertain details for the query, ask for further information if clarification is required to work on the task in question, or simply perhaps proceed to update task status to enable progress tracking by project team and leadership.

Next, **DEV1** hence accessed the Atlassian portal to get information on the task received as illustrated figure 6.10:


Create new code repository for system

Add a description...

Activity

PM1 51 seconds ago

Please create a public repo to attract outside contributors

Add a comment...

Status: To Do

Assignee: DEV1

Labels: None

Reporter: PM1

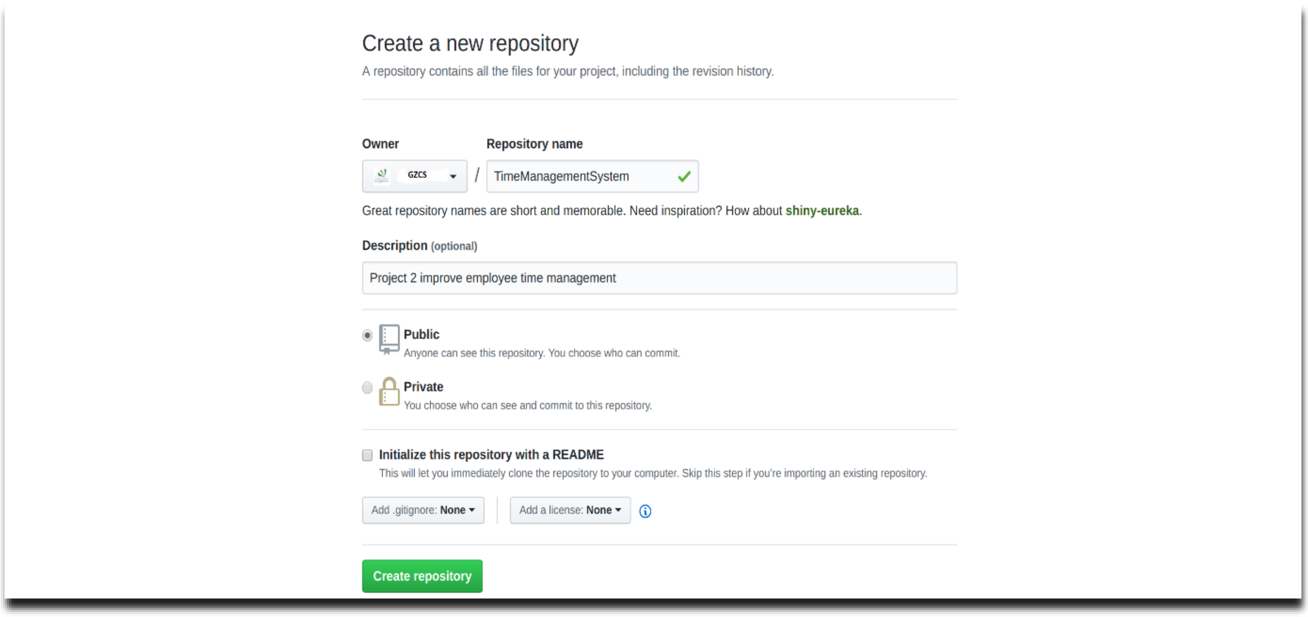
Created 1 hour ago

Updated 1 hour ago

Figure 6.10 JIRA task examination

From the request transmitted via Slack and depicted in the previous two figures, **DEV1** upon further verification on the JIRA project management Kanban board understood that the creation of a new public code repository was required and that this task had been assigned to him by **PM1.**

The organisational standard for GZ Consulting Services being that all company source code be hosted on GITHUB.com, **DEV1** navigated to the cloud Source Control Management System (SCM) online portal for creation of a GIT repository to host source code. He proceeded to do so as required by the request sent and ensured that this GIT artefact was publicly accessible. This project would allow outside contributions unless another request was later sent by management to make the project private.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: GZCS / Repository name: TimeManagementSystem ✓

Great repository names are short and memorable. Need inspiration? How about **shiny-eureka**.

Description (optional): Project 2 improve employee time management

[✓ **Public**] Anyone can see this repository. You choose who can commit.

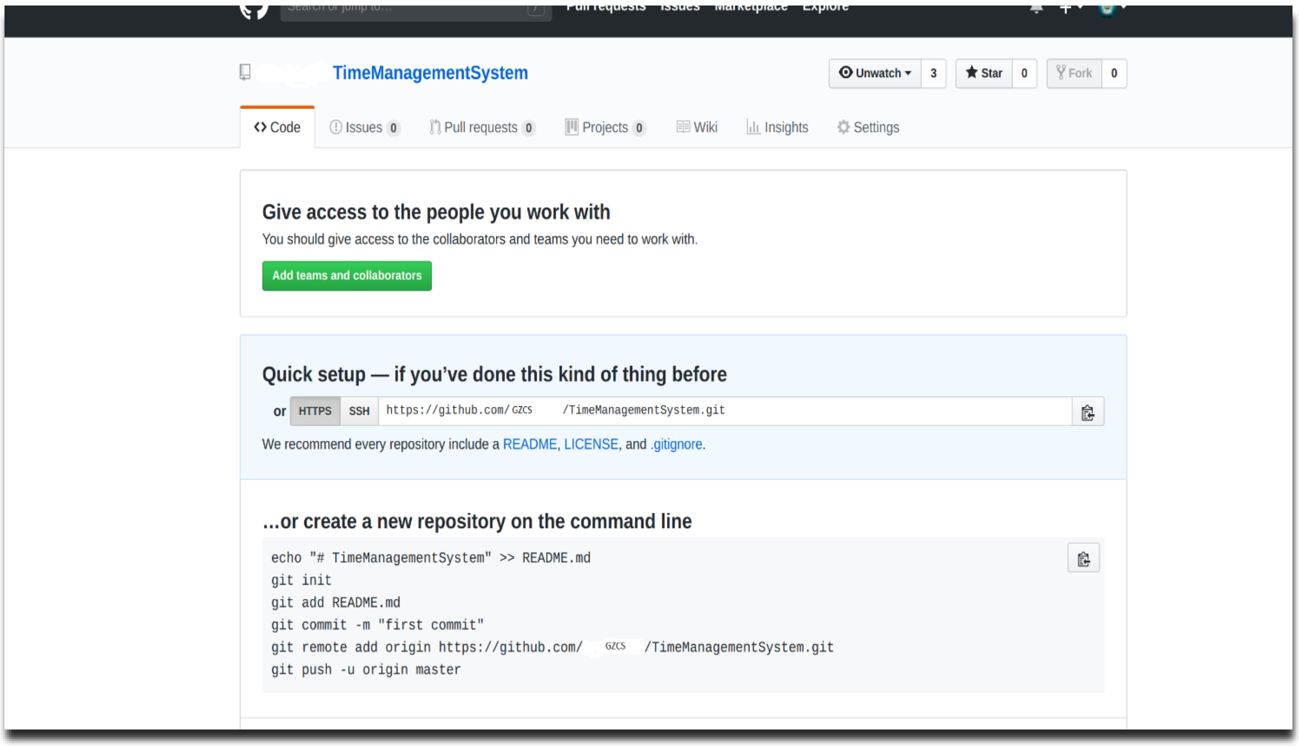
Private You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Figure 6.11 Creation of new source code repository on GITHUB.com

The screenshot above illustrates the creation of GIT source code repository by **DEV1** under the GZCS organisation on GITHUB which is accessible on the public domain as requested by **PM1**. Figure 6.12 depicts options offered by GITHUB.com to work with the created repository in question once SCM artefacts have successfully been created.



Give access to the people you work with

You should give access to the collaborators and teams you need to work with.

Quick setup — if you've done this kind of thing before

or HTTPS SSH `https://github.com/GZCS/TimeManagementSystem.git`

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# TimeManagementSystem" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/GZCS/TimeManagementSystem.git
git push -u origin master
```

Figure 6.12 Source code GIT artefact after creation on GITHUB.com

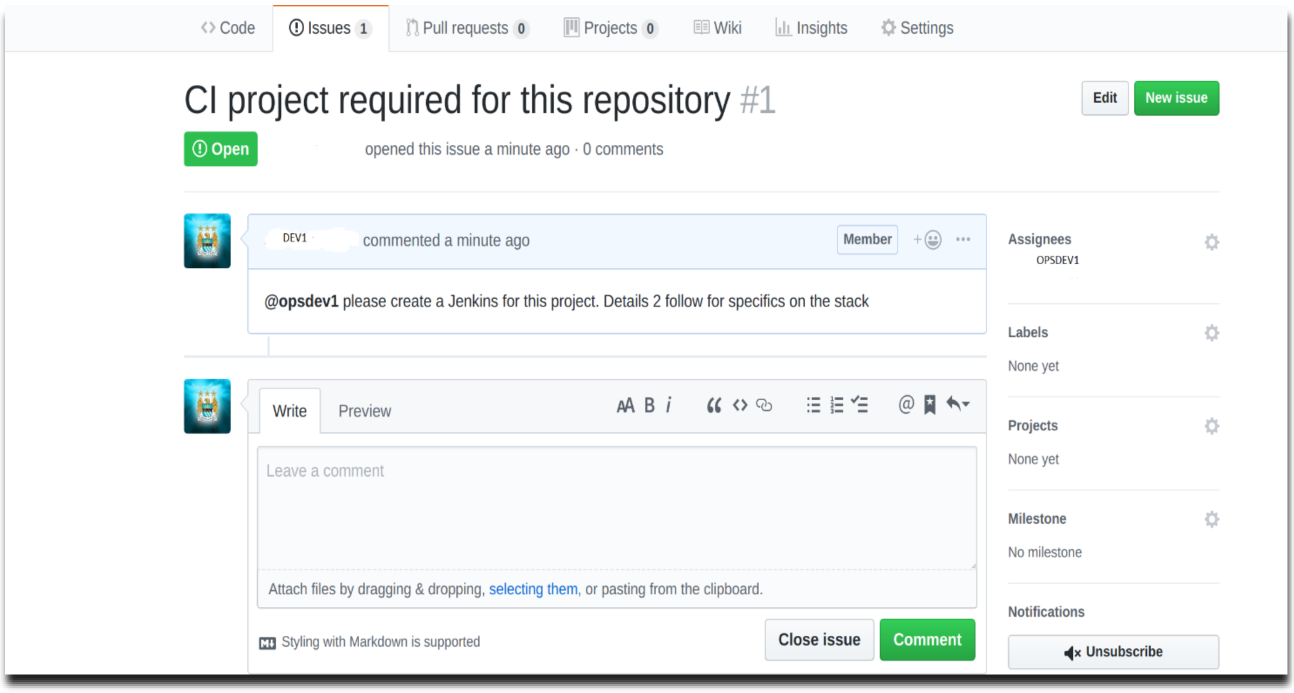
Although there had been no clear instructions regarding programming languages to be used for the code repository in question, **DEV1** could anticipate that regardless the technology stack decided for this project, there would be a requirement to provide code building and deployment facilities down the line. In such a case, there are typically two main options available to pro-actively prepare build and deployment activities which are needed as code repositories are progressively updated:

- Notify **PM1** that a task needs to be created on the JIRA board for configuration to be setup so that the code is seamlessly built and deployed to required operational environments such as Development, Quality Assurance or Production.
- **DEV1** could otherwise create this task himself and assign it to a resource without waiting for **PM1** to intervene.

When the first option is chosen, the task assignment workflow restarts from the point where **PM1** would logon to the JIRA portal, create a new ticket for build and deployment activities and assign it to a new resource which would then act upon it. If the latter option is selected, **DEV1** can create this ticket by using either JIRA or the work assignment facility provided by GITHUB.com. Although this can have the disadvantage of not providing immediate visibility on JIRA if the portal has not been setup accordingly, at GZCS, engineering staff have the latitude to use either or. During the use case observed, **DEV1** opted to create a GITHUB issue for a colleague whose speciality is the continuous integration and deployment (CI/CD) space.

In this scenario, the engineer in question identified as **OPSDEV1** received a work request from his colleague **DEV1** to start working on continuous build and deployment workflow for the source code repository**.**

Below is an overview of the request received by **OPSDEV1** via GITHUB.com to handle CI requirements for this scenario:



CI project required for this repository #1

opened this issue a minute ago · 0 comments

DEV1 commented a minute ago

@opsdev1 please create a Jenkins for this project. Details 2 follow for specifics on the stack

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Assignees: OPSDEV1

Labels: None yet

Projects: None yet

Milestone: No milestone

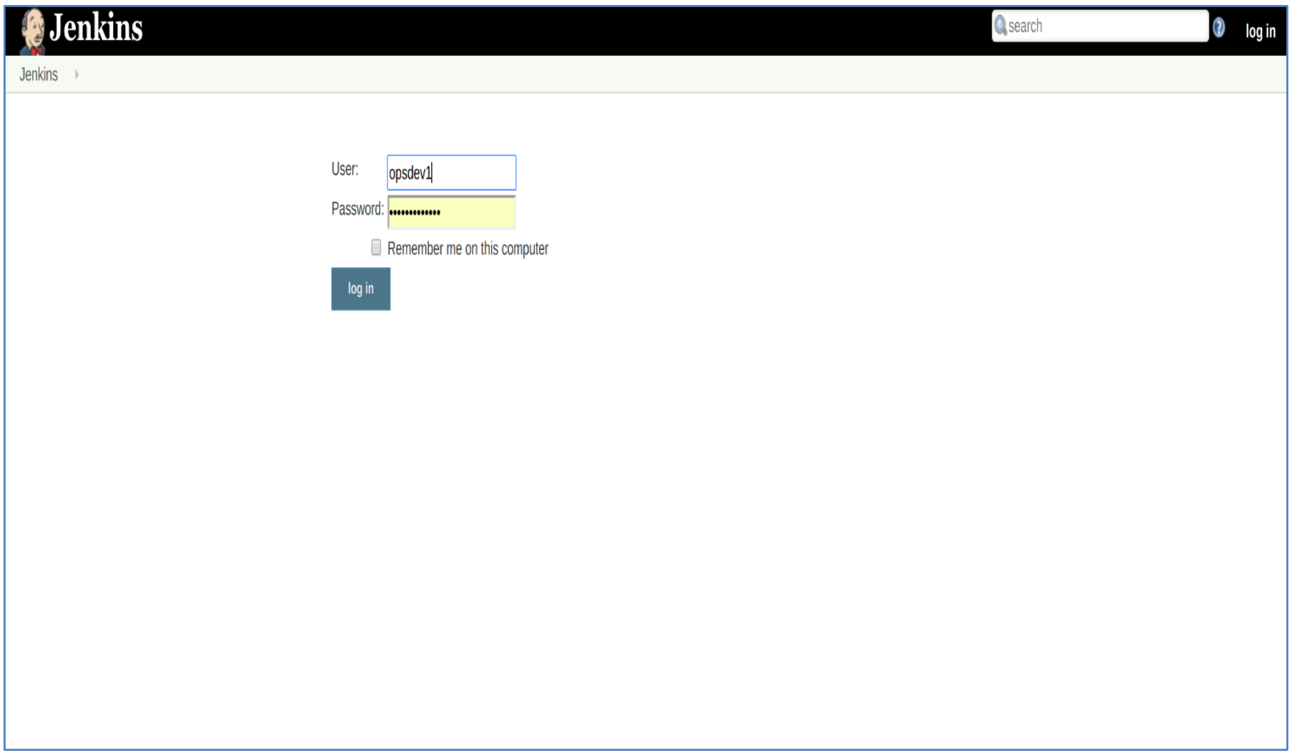
Figure 6.13 GITHUB issue examination

Having received the request above with admittedly very little technical detail, **OPSDEV1** was nonetheless able to at least create a skeleton build and deployment pipeline for the code repository in question.

To achieve effective CI/CD practices and workflows, agile engineering teams typically leverage enabling tools to automate code builds as well as related deployments to test, quality assurance and production environments. Such teams are often attracted to CI/CD because projects which leverage it have been demonstrated to harbour fewer bugs and not be as error prone when automated properly (Fowler 2006, p. 12).

Consequently, engineers nowadays rely on enabling tools which additionally improve quality assurance by creating workflows which are only completed once unit and integration test cases are successfully executed; this is done so that appropriate code versions are deployed at the right time for the right target environments (Juopperi 2017, p. 3).

As discussed in earlier chapters, Jenkins and Amazon Web Services' CodeBuild are such enabling tools that comprise the primary systems utilised at GZ Consulting Services for CI/CD activities. For the current use case, all that was required by **OPSDEV1** to attend the task assigned, was to logon to the Jenkins portal and configure a build project automatically hooked to the freshly created GITHUB repository by **DEV1**.



Jenkins

search

log in

Jenkins

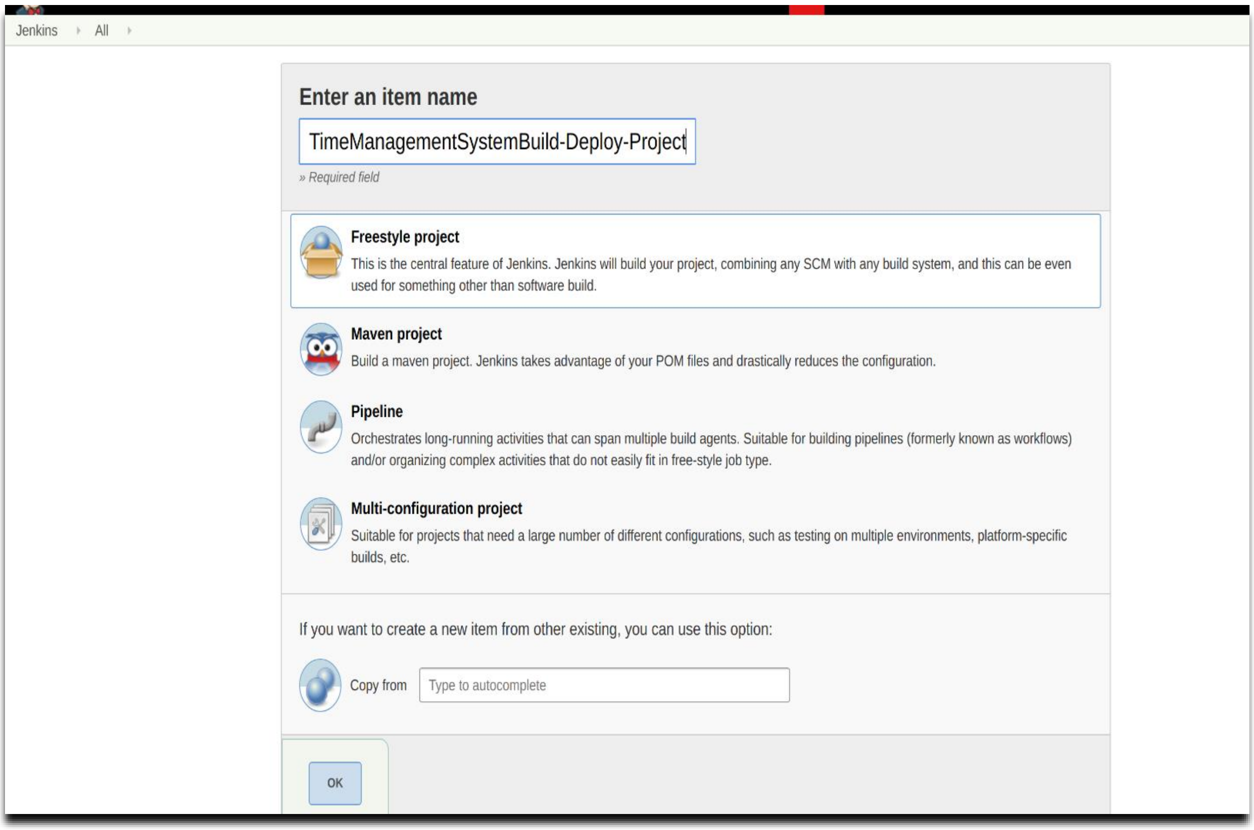
User: opsdev1

Password: ••••••••••••

Remember me on this computer

log in

Figure 6.14 OPSDEV1 Jenkins login portal



Jenkins › All ›

Enter an item name

TimeManagementSystemBuild-Deploy-Project

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project

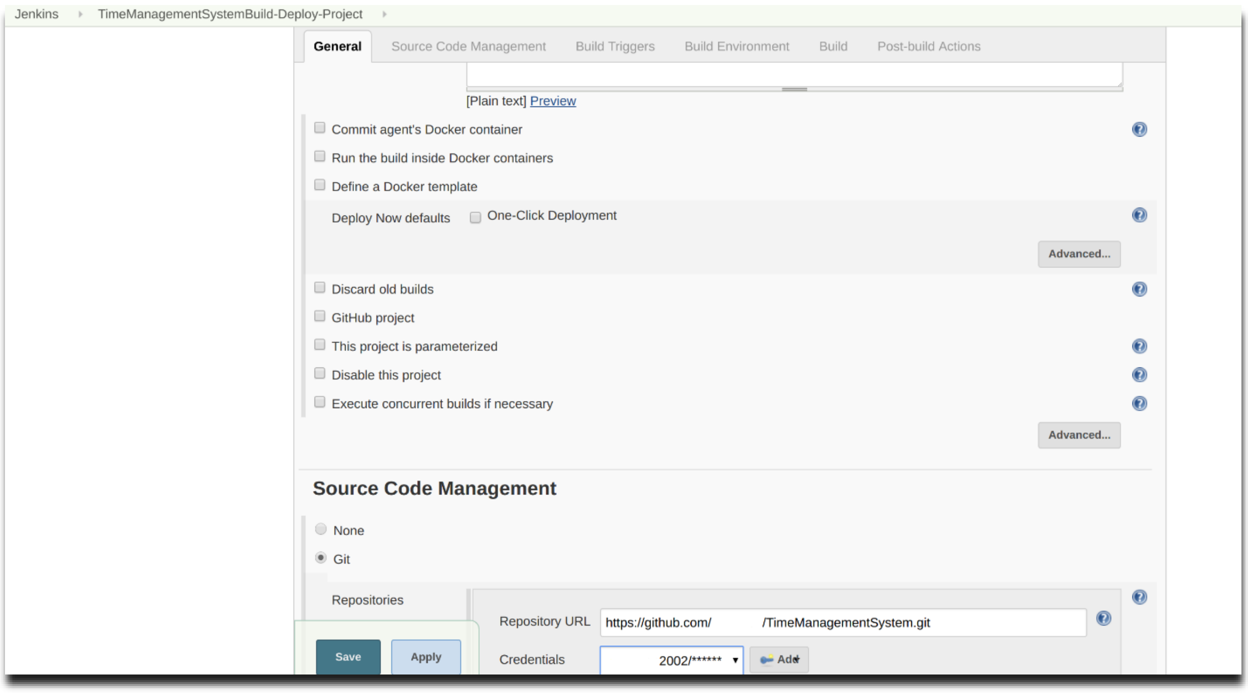
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

If you want to create a new item from other existing, you can use this option:

Copy from Type to autocomplete

OK

Figure 6.15 Creation of Jenkins build and deploy pipeline


Jenkins › TimeManagementSystemBuild-Deploy-Project ›

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

[Plain text] Preview

- [ ] Commit agent's Docker container
- [ ] Run the build inside Docker containers
- [ ] Define a Docker template

Deploy Now defaults [ ] One-Click Deployment

Advanced...

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This project is parameterized
- [ ] Disable this project
- [ ] Execute concurrent builds if necessary

Advanced...

Source Code Management

None [✓ Git]

Repositories

Repository URL: https://github.com/ /TimeManagementSystem.git

Credentials: 2002/******

Figure 6.16 Jenkins build and deploy pipeline configuration from GIT SCM

After initial configuration, the Jenkins project depending on whether it was setup with correct information or not; is able to continuously poll the code repository for changes. In such occurrences, it can checkout source code, build the selected artefacts and ultimately deploy them to correct environments specified during configuration. Showing the results of these builds for the use case observed was not primordial. What needs to be noted though is that the outcome of these builds once triggered was transparent to both **PM1** and **DEV1** unless explicitly configured. Giving them a continuous view and progress updates on the process would have required prior customization of communications channels used at GZCS on JIRA, Slack as well as GITHUB and email interfaces. In the next section, a summarized analysis of the scenario described during portion 6.3.1 of the current chapter.

6.3.3 Pre-Implementation observational synopsis

Conducting data gathering in the observational phase of this research revolved around exploring daily tasks of employees during the case study at GZ Consulting Services.

Watching and documenting some of these daily processes was conducted to fit in with the identified activities of this thesis' activity theory informed framework. The daily processes to be observed included engineering (software development and DevOps) as well as project management related activities (creation of new projects, resources assignments amongst others).

The researcher attempted observation of these activities prior to the implementation of the proposed agile oriented IS prototype. Figure 6.17 illustrates the scenarios observed prior to this implementation from a leadership and engineering viewpoint:

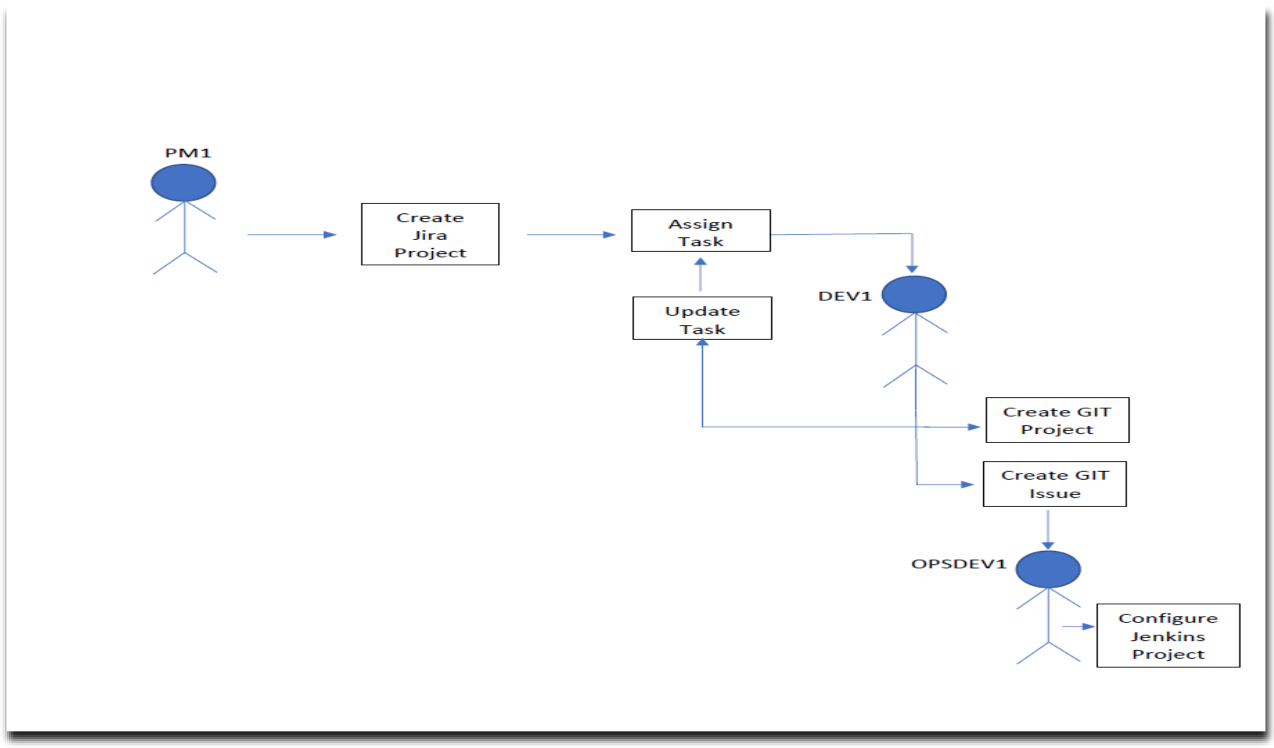


Figure 6.17 Pre-Implementation scenario observational scenarios

Scenarios observed prior to the research IS prototype concerned three GZ Consulting Services staff members:

- A project manager identified as **PM1.**
- An engineer focused on the code development space identified as **DEV1.**
- An engineer specialized in the Continuous Integration and Deployment space identified as **OPSDEV1.**

In scenarios observed at this stage, project managers essentially interacted with engineers via the corporate JIRA portal for project management related activities. This was done with regards to tasks such as creating new business projects based on feedback from customer and executive requirements, dealing with project resources allocation, and assigning tasks to engineers.

In this scenario, for engineers to whom tasks are typically assigned by leadership (for example, **PM1** assigned tasks to **DEV1**) via JIRA, interactions with their project leader is done upstream via the same JIRA portal to update task statuses and ask for clarifications or further information where required. Nonetheless, for downstream communication to engineering colleagues, usage of other channels such as JIRA, GIT, Slack, and email was at times also noticed. For instance, in the use cases observed, **DEV1** received a task via one channel (JIRA) and assigned a related request to **OPSDEV1** via another such as GITHUB's ticketing facility.

This situation, considered from an engineering and project leadership perspective, is not uncommon for organisations which have opted to embrace agile and DevOps approaches to running their projects. When embarking on a quest to achieve delivery of quality products to markets, companies and their project teams need to shorten their production lifecycles with increased levels of automation (Christof *et al.* 2016, p. 95).

From the observational data gathered pre-implementation at GZ Consulting Services, it appeared that despite high usage of agile and DevOps related technology tooling other issues arose.

In effect, notifications where automatically pushed to relevant resources depending on channel used by requester whether through Slack, JIRA, GIT or plain email; consequently, there was still a need to login and navigate to each system's portal for access to information. Additionally, at certain points of the work assignment chain, some parts of the processes surrounding engineering activities were completely opaque to project leadership. **PM1** for instance had no way to physically ensure that a mundane task such as the creation of code repositories was executed. As the project went along, it was hence difficult in this scenario to track work velocity on bug reports or new features requests at an engineering level.

DEV1 from his point of view, resented the fact that too much of his time was spent trying to convince project leadership that some work was already completed. An example of this occurred during an argument with **PM1** about emails sent by other contributors. In this case, a few external engineers complained about not being able to push code to a non-existent project repository. Having to deal with a visibly upset **PM1, DEV1** bemoaned the fact that there was no way to convince his colleague beyond a shadow of doubt that this task had been completed despite status updates on the JIRA Kanban board and in verbal form during scrum sessions.

From a somewhat neutral observer's perspective, the apparent problem in this scenario was a perceived opacity between leadership and engineering staff. This was in fact caused by the apparent lack of integration between systems in use at the firm. In practice, with today's modern tools, many activities can be performed in agile projects by personnel which need not be professional engineers or particularly technically gifted. For basic requirements at project inception stages such as creating source code repositories, it could perhaps be preferable that tasks which could be handled by a project manager for example not be hampered by false technical barriers. In the creation of code repository during project inception scenario, an argument could be made that if **PM1** had access to the GIT system, he could have simply run the commands manually or via the GITHUB portal to complete this simple task. The current research objectives dictated such a manual approach be rather automated so that even non-technical team members could contribute in some respects.

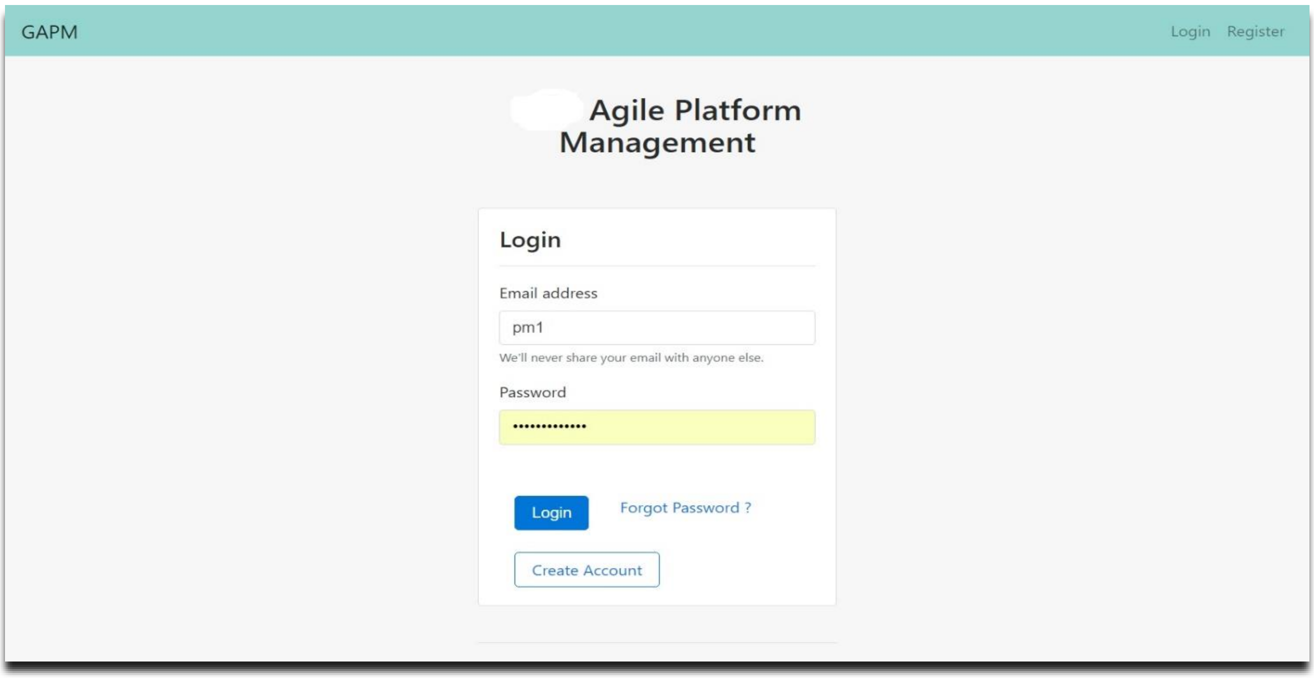
Furthermore, the issue of reliable versions of truth arose. Where could **PM1** have access to reliably accurate project information when some items where only visible on other systems used at GZCS such as Slack, GITHUB or Jenkins by subject matter experts?

These are the kinds of shortcomings the proposed IS prototype described in Chapter Five needed to address. To this point, the next section proceeds to describe a project implementation scenario after this thesis' agile oriented IS was deployed for usage by GZ Consulting Services staff.

6.3.4 GZCS Software Project inception post-implementation: Leadership perspective

Observational data gathered after implementation of the proposed IS prototype described in Chapter Five follows:

In the first use case, **PM1** needed to initiate a project creation workflow from scratch after having been on onboarded as a system user on the proposed IS prototype. A key objective was to try ensuring project leadership rely only on the platform provided for this study during new project inception. Some GZ Consulting Services project information data was loaded onto the system to provide an intuitive look and feel for new users; but most information displayed at this staged was automatically synced from the agile tools in use at the firm to which this IS prototype integrates. To address the problem of having to log on to multiple systems for simple project inception, the prototype IS proposed a workflow which via one centralized hub, could create all the basics relevant artefacts in JIRA as well as on the GIT source code control management system.


GAPM Login Register

Agile Platform Management

Login

Email address

pm1

We'll never share your email with anyone else.

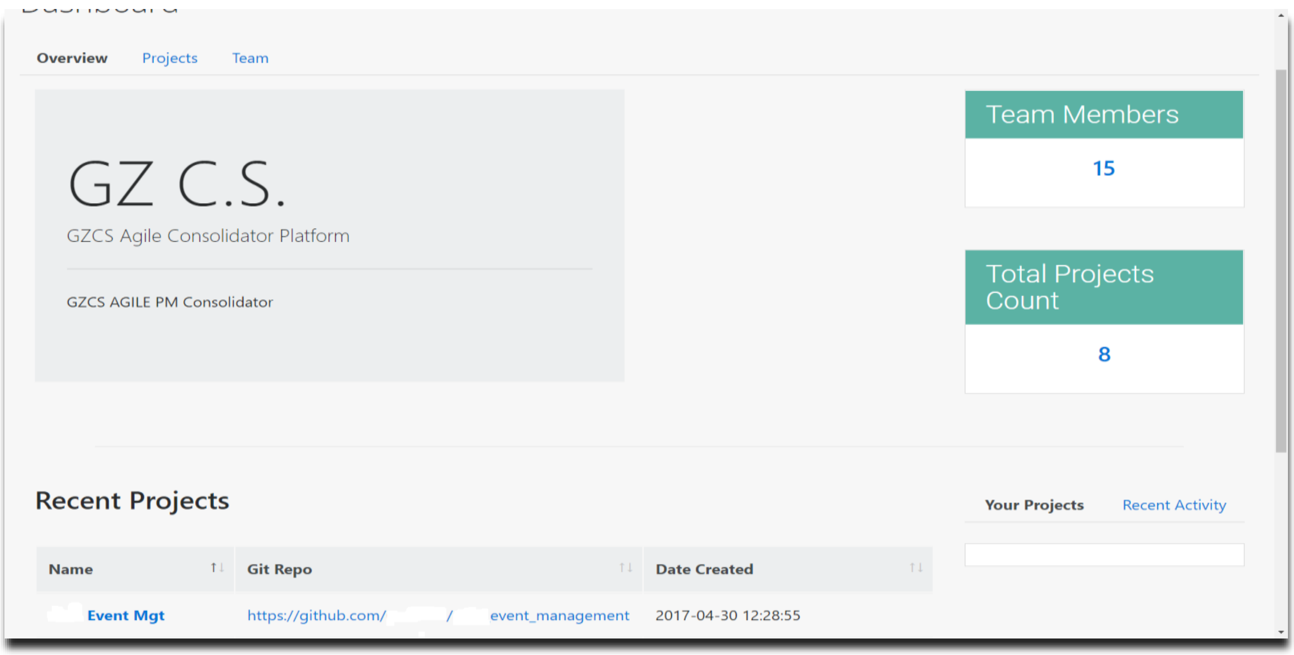
Password

•••••••••••••

Login Forgot Password ?

Create Account

Figure 6.18: GZ Agile PM Consolidator login portal


Overview Projects Team

GZ C.S.

GZCS Agile Consolidator Platform

GZCS AGILE PM Consolidator

Team Members

15

Total Projects Count

8

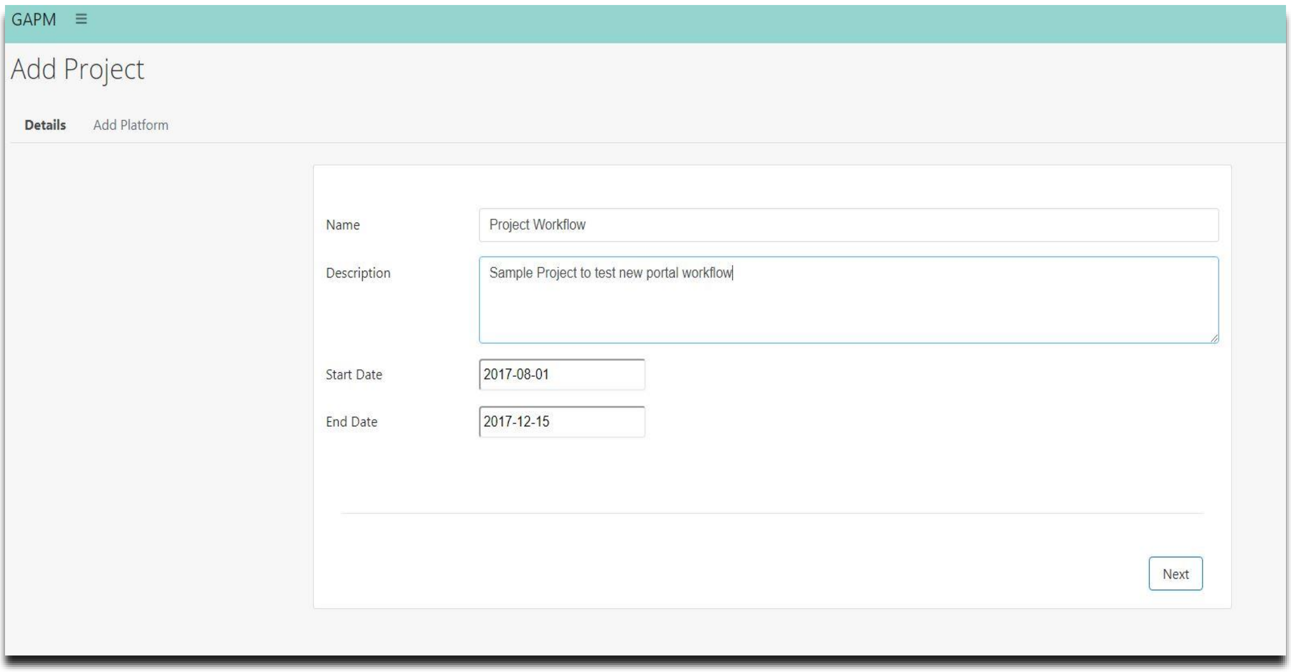
Recent Projects

Your Projects Recent Activity

Name	Git Repo	Date Created
Event Mgt	https://github.com/ / event_management	2017-04-30 12:28:55

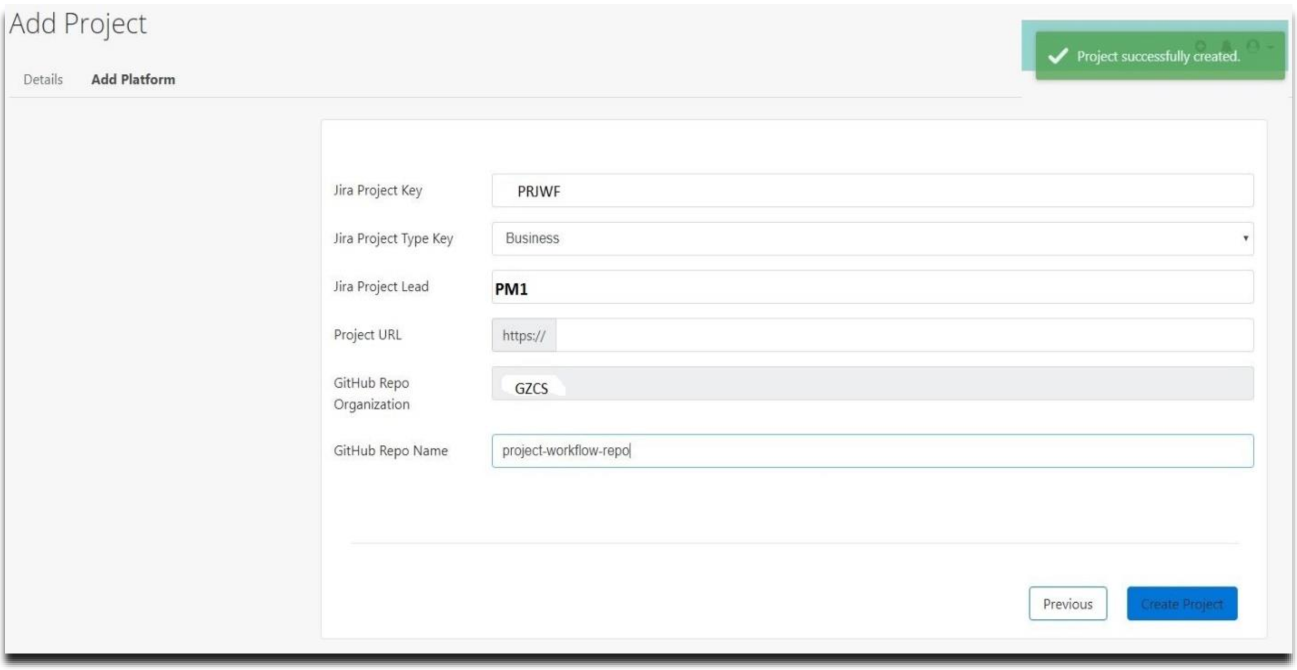
Figure 6.19: GZ Agile Consolidator Portal home page

After having been authenticated to the prototype portal as depicted in screenshots above, **PM1** was directed to his dashboard were basic organisational project information is available to view. In the use case where he needed to create a new project, a project workflow needed to be initiated as to generate all relevant artefacts in integrated agile/DevOps systems used at GZ Consulting Services. The workflow was triggered by clicking the plus button circled in red at the top right hand of his dashboard. After this, the user was transferred from the main portal to the next project creation page:



Name: Project Workflow
Description: Sample Project to test new portal workflow
Start Date: 2017-08-01
End Date: 2017-12-15

Figure 6.20: Basic project information entry



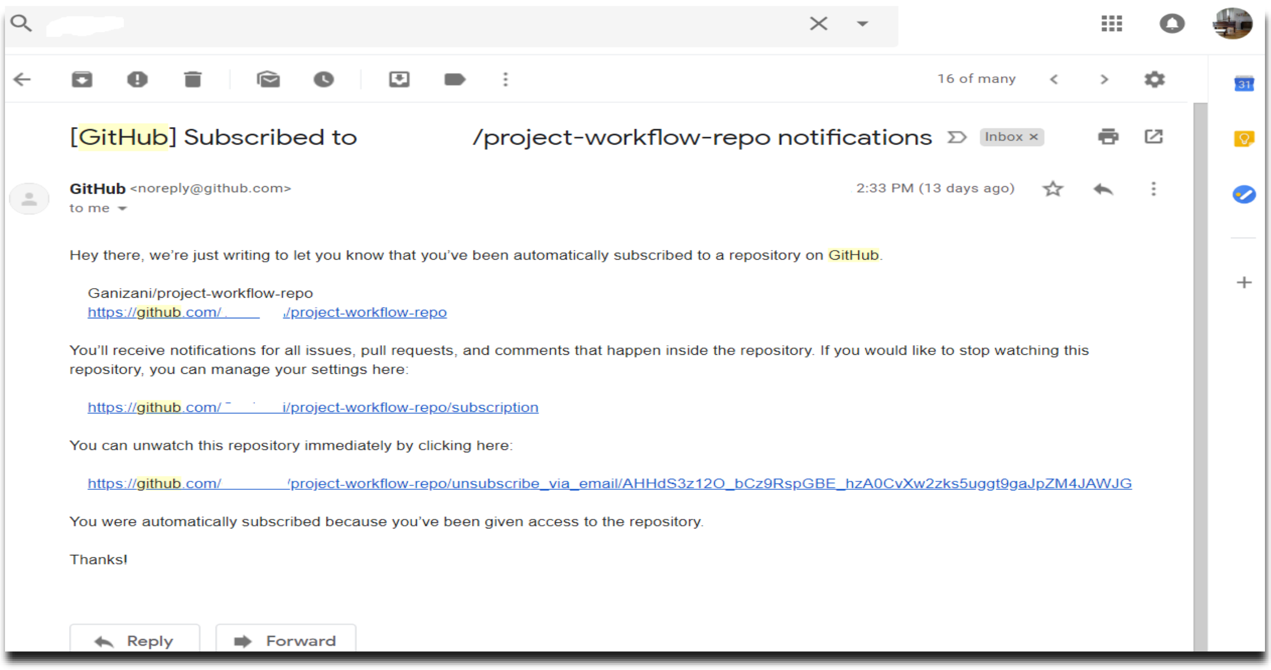
Project successfully created.

Jira Project Key: PRJWF
Jira Project Type Key: Business
Jira Project Lead: PM1
Project URL: https://
GitHub Repo Organization: GZCS
GitHub Repo Name: project-workflow-repo

Figure 6.21: Successful project creation

In the screenshots above, **PM1** first entered basic project information on the portal with details such as the project name, a description of its purposes as well as the projected start and end dates for the new endeavour. All the basic information was stored in the prototype's internal database. After this was done, **PM1** was directed to the next page where he was prompted to enter further information related to details which needed to be used and synced across the organisation's SCM and project management systems: GITHUB and JIRA.

When done correctly, both engineering staff as well as project leadership would be on the same page in terms of possessing the same information after the inception of new projects. The data entered at this stage needed to respect third party system rules for the creation of relevant artefacts (for instance, JIRA will not let a user create a project with key which exceeds 8 characters or with a project name already present). Validation rules were consequently built into the proposed IS prototype to ensure data integrity across GZCS's internal systems. Once a project is created successfully in this scenario, engineers will typically receive a GITHUB notification of the code project created in the SCM system as below:



[GitHub] Subscribed to /project-workflow-repo notifications

GitHub <noreply@github.com>
to me

2:33 PM (13 days ago)

Hey there, we're just writing to let you know that you've been automatically subscribed to a repository on GitHub.

Ganizani/project-workflow-repo
https://github.com/ /project-workflow-repo

You'll receive notifications for all issues, pull requests, and comments that happen inside the repository. If you would like to stop watching this repository, you can manage your settings here:

https://github.com/ i/project-workflow-repo/subscription

You can unwatch this repository immediately by clicking here:

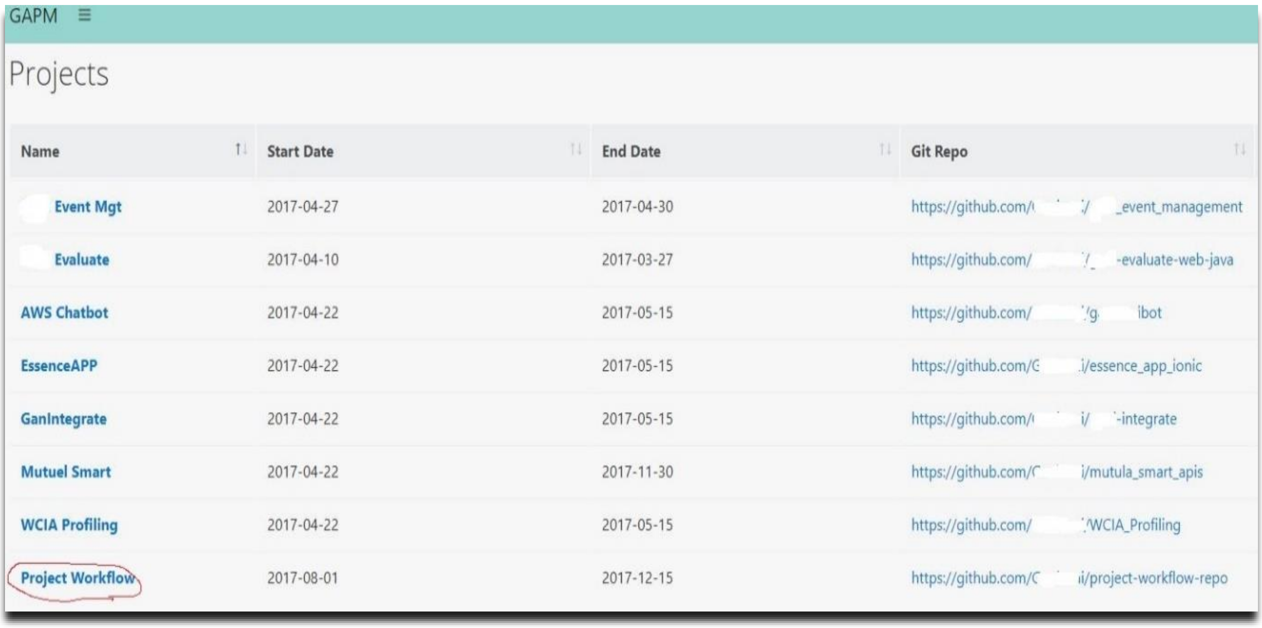
https://github.com/ /project-workflow-repo/unsubscribe_via_email/AHHdS3z12O_bCz9RspGBE_hzA0CvXw2zks5uggt9gaJpZM4JAWJG

You were automatically subscribed because you've been given access to the repository.

Thanks!

Figure 6.22: GIT SCM success notification

When navigating back to the prototype IS portal, the internal project artefact was now shown on the GZ Agile central portal:

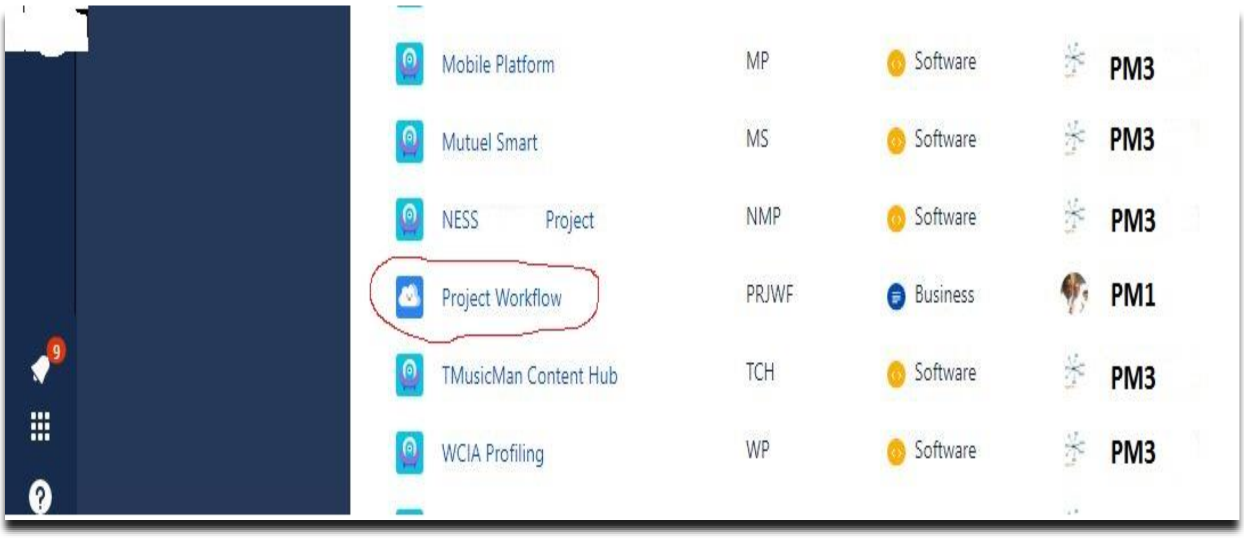

GAPM

Projects

Name	Start Date	End Date	Git Repo
Event Mgt	2017-04-27	2017-04-30	https://github.com/ / _event_management
Evaluate	2017-04-10	2017-03-27	https://github.com/ /_ -evaluate-web-java
AWS Chatbot	2017-04-22	2017-05-15	https://github.com/ /g ibot
EssenceAPP	2017-04-22	2017-05-15	https://github.com/G i/essence_app_ionic
GanIntegrate	2017-04-22	2017-05-15	https://github.com/ i/ -integrate
Mutuel Smart	2017-04-22	2017-11-30	https://github.com/C i/mutula_smart_apis
WCIA Profiling	2017-04-22	2017-05-15	https://github.com/ /WCIA_Profiling
Project Workflow	2017-08-01	2017-12-15	https://github.com/C i/project-workflow-repo

Figure 6.23: GZ Agile Consolidator central project page

PM1 was then able to check if the running the prototype's workflow resulted in the successful generation of related artefacts in other internal system instances at used GZ Consulting Services.

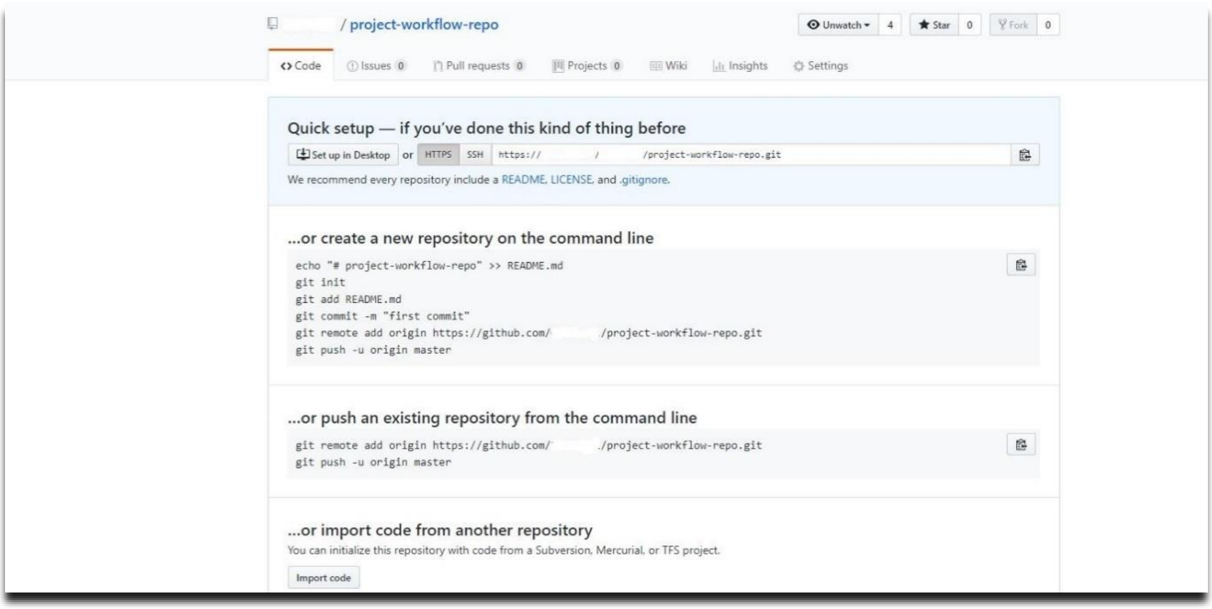


Mobile Platform	MP	Software	PM3
Mutuel Smart	MS	Software	PM3
NESS Project	NMP	Software	PM3
Project Workflow	PRJWF	Business	PM1
TMusicMan Content Hub	TCH	Software	PM3
WCIA Profiling	WP	Software	PM3

Figure 6.24: JIRA project artefact generated by GZ-AgilePM Consolidator

From the screenshot above, the artefact generated from GZ-AgilePM Consolidator is circled in red. The difference between projects synchronised between the IS prototype central portal and items created directly through JIRA is the cloud icon next to the project name. This icon differentiates projects created through the JIRA user interface and third-party applications like this study's prototype which leverage vendor provided Application Programming Interfaces

(APIs). Similarly, engineers subscribed to the organisational GIT repositories were then also able to confirm whether the requested repository on the SCM system was successfully created via the IS prototype portal:



Quick setup — if you've done this kind of thing before

```
https://          /          /project-workflow-repo.git
```

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# project-workflow-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/          /project-workflow-repo.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/          ./project-workflow-repo.git
git push -u origin master
```

...or import code from another repository

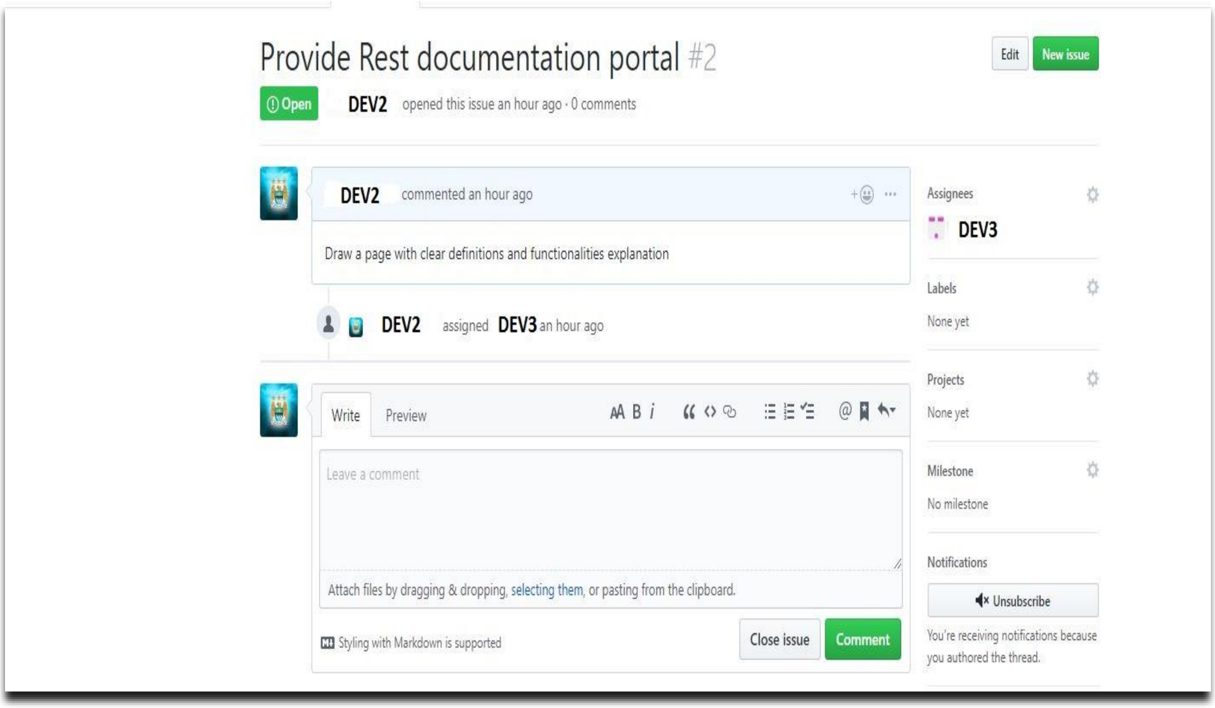
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Figure 6.25: SCM project artefact generated by GZ-AgilePM Consolidator

6.3.5 GZCS Software Project execution post-implementation: Engineering viewpoint

From an engineering point of view, at GZ Consulting Services, tasks or requests for bug fixes and enhancements can be logged through the GIT SCM portal by any employee who has the correct set of permissions. Software engineers can also get new tickets assigned to them for additional features and components by project managers via the JIRA portal.

Consider the following GIT issue logged on the SCM portal by an engineer identified as **DEV2** for a colleague **DEV3** to ensure a web page is designed for API definitions:


Provide Rest documentation portal #2

Open DEV2 opened this issue an hour ago · 0 comments

DEV2 commented an hour ago

Draw a page with clear definitions and functionalities explanation

DEV2 assigned DEV3 an hour ago

Assignees: DEV3

Labels: None yet

Projects: None yet

Milestone: No milestone

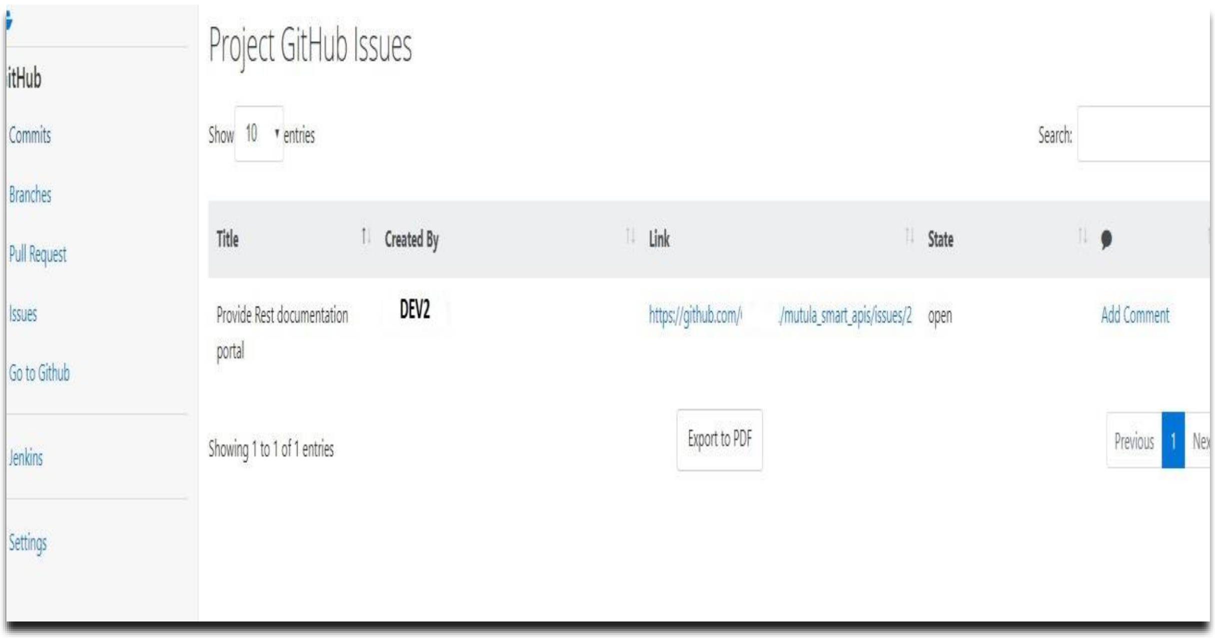
Notifications

You're receiving notifications because you authored the thread.

Edit
New issue
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Close issue
Comment
Unsubscribe

Figure 6.26: Request logged via SCM Portal

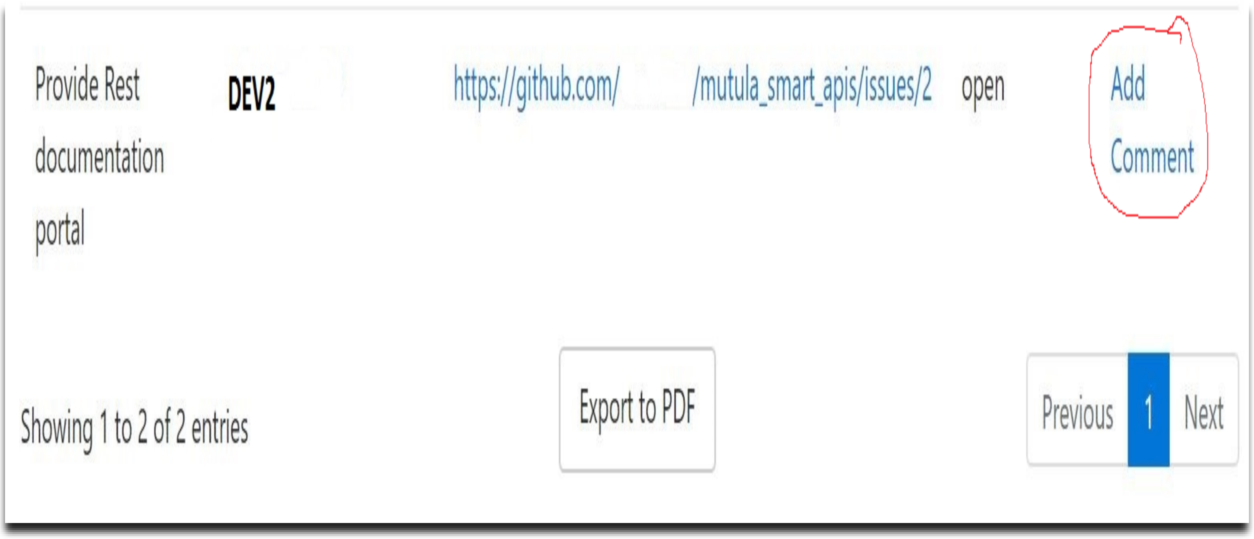
Once a staff member be it for project management reasons or the engineer in question logs onto the current research's proposed IS tool after having been notified of the request, he is able to see the data which was seamlessly transmitted via internal API calls as below:


Project GitHub Issues

Title	Created By	Link	State	
Provide Rest documentation portal	DEV2	https://github.com/ /mutula_smart_apis/issues/2	open	Add Comment

Showing 1 to 1 of 1 entries

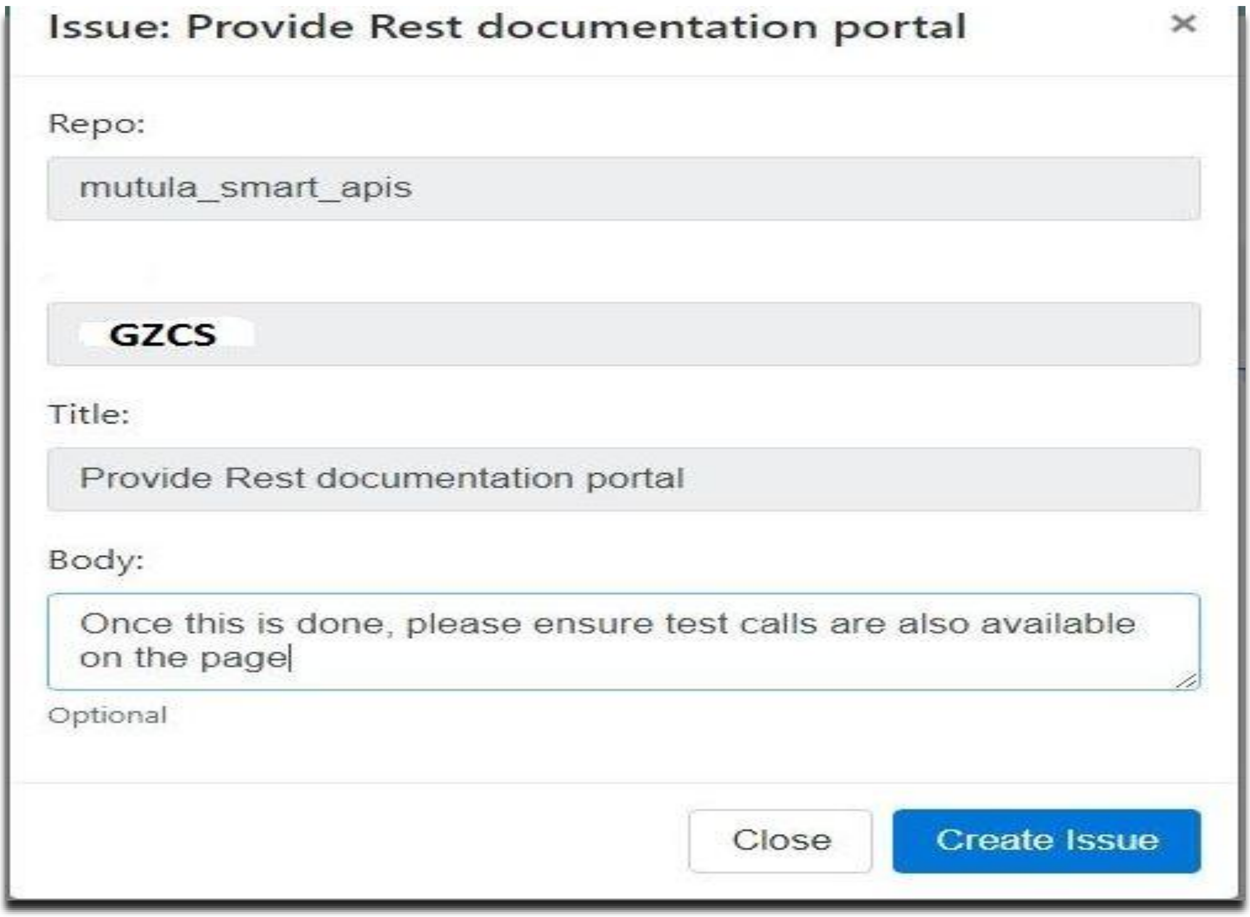
Figure 6.27: GZ-AgilePM Consolidator synchronised data from SCM


Provide Rest documentation portal DEV2 https://github.com/ /mutula_smart_apis/issues/2 open Add Comment

Showing 1 to 2 of 2 entries

Figure 6.28: New linked issue triggered via click on comment button

Further related issues can then be created via the "Add Comment" hyperlink circled in red above. The IS tool subsequently synchronised this data back to the SCM system as below:


Issue: Provide Rest documentation portal

Repo:
mutula_smart_apis

GZCS

Title:
Provide Rest documentation portal

Body:
Once this is done, please ensure test calls are also available on the page

Optional

Figure 6.29: Creation of new related issue to SCM via IS portal

After the page the details are entered in regard to related new work requests, the user clicks on the "Create Issue" button and is presented with a successful prompt if the information is correctly synchronised to the cloud Source Control Management System in GITHUB.

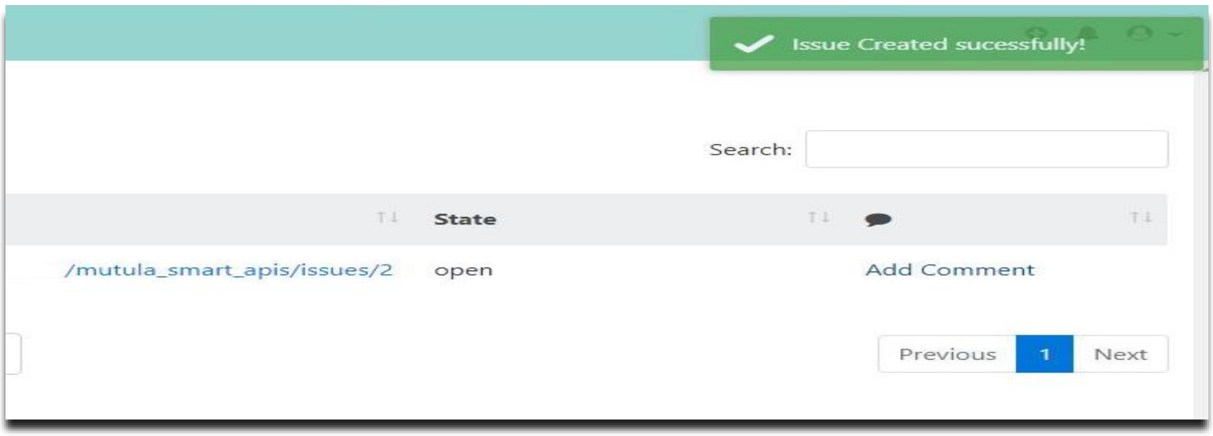
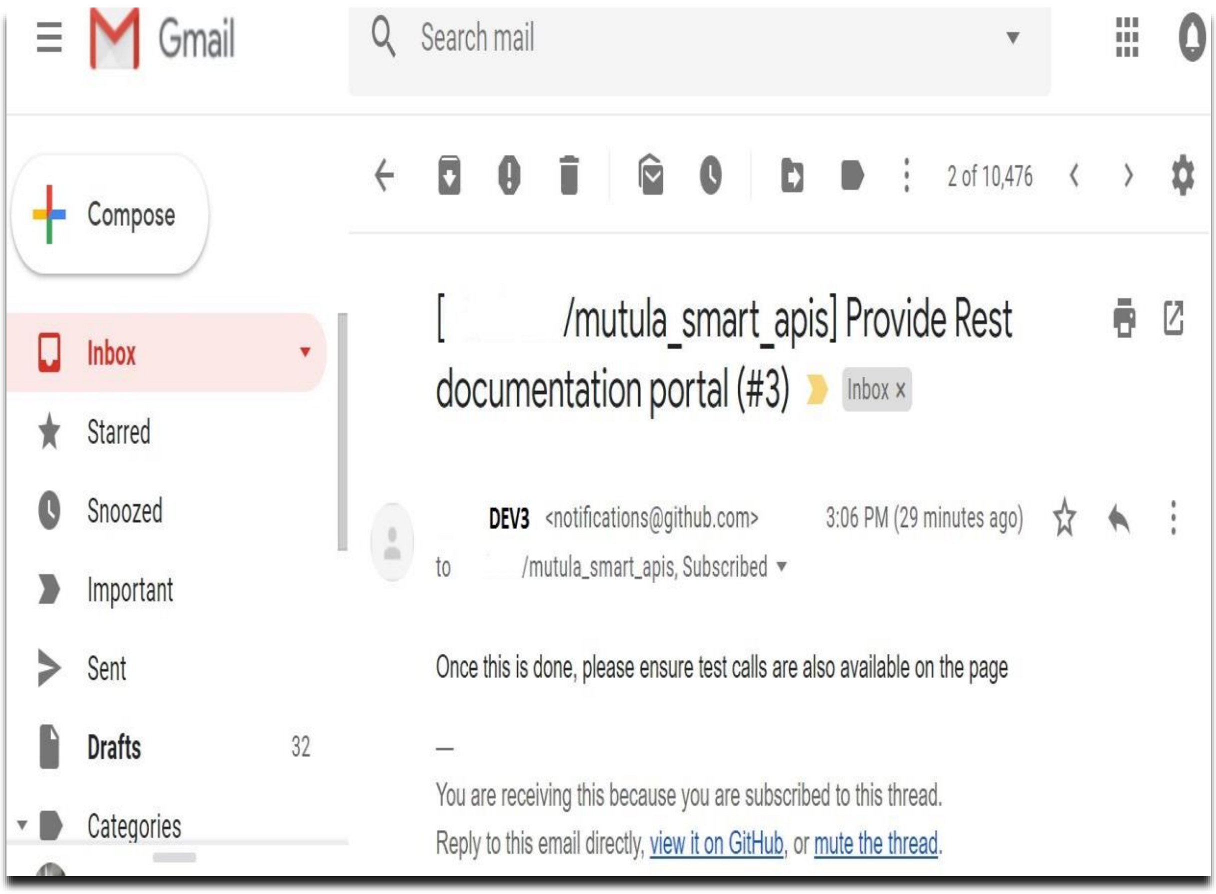


Figure 6.30: Successful creation of new issue via portal

Colleagues are then able to receive synchronised notifications via messaging communication channels:



[/mutula_smart_apis] Provide Rest documentation portal (#3) Inbox x

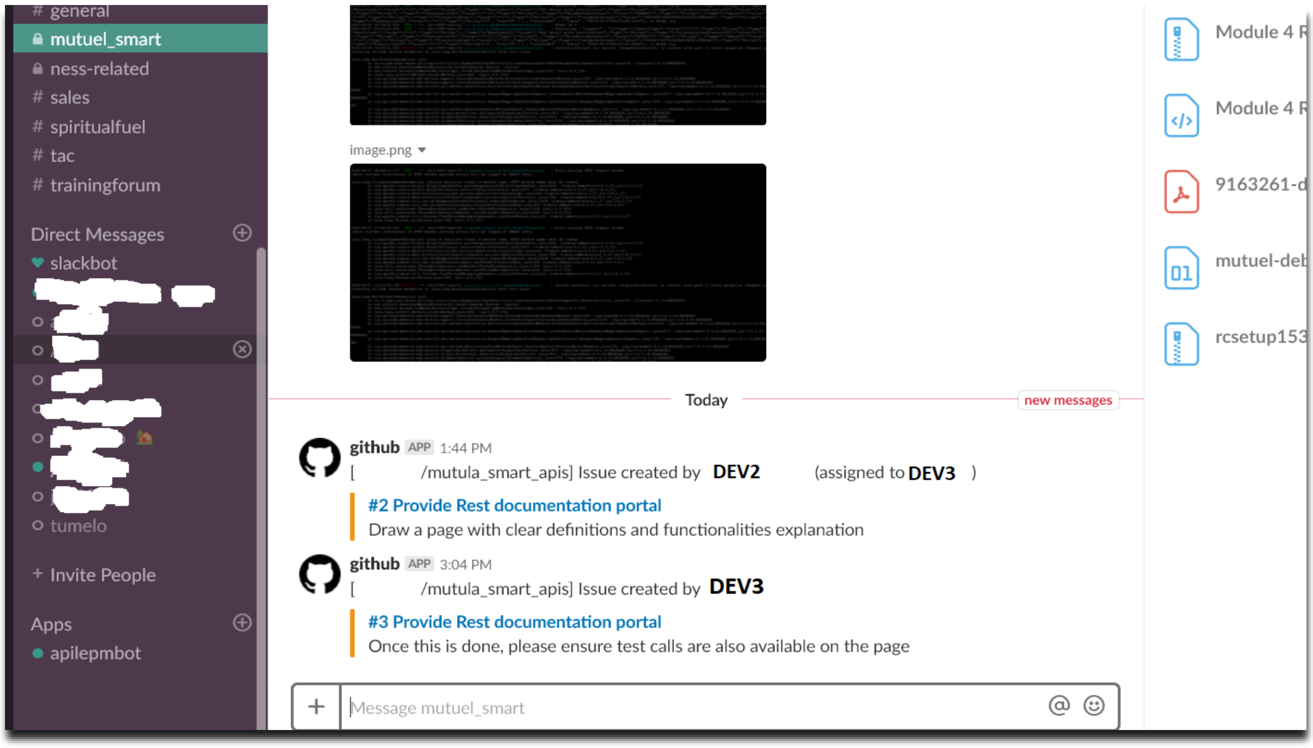
DEV3 <notifications@github.com> 3:06 PM (29 minutes ago)
to /mutula_smart_apis, Subscribed

Once this is done, please ensure test calls are also available on the page

—
You are receiving this because you are subscribed to this thread.
Reply to this email directly, view it on GitHub, or mute the thread.

Figure 6.31: SCM issue email notification triggered by IS prototype

To illustrate the almost immediate nature of data synchronisation facilitated by GZ-AgilePM Consolidator. Figure 6.31 illustrates an email sent by the SCM system after creation of a code related issue through the IS prototype; the screenshot below describes a similar notification sent via the organisational ChatOps system:

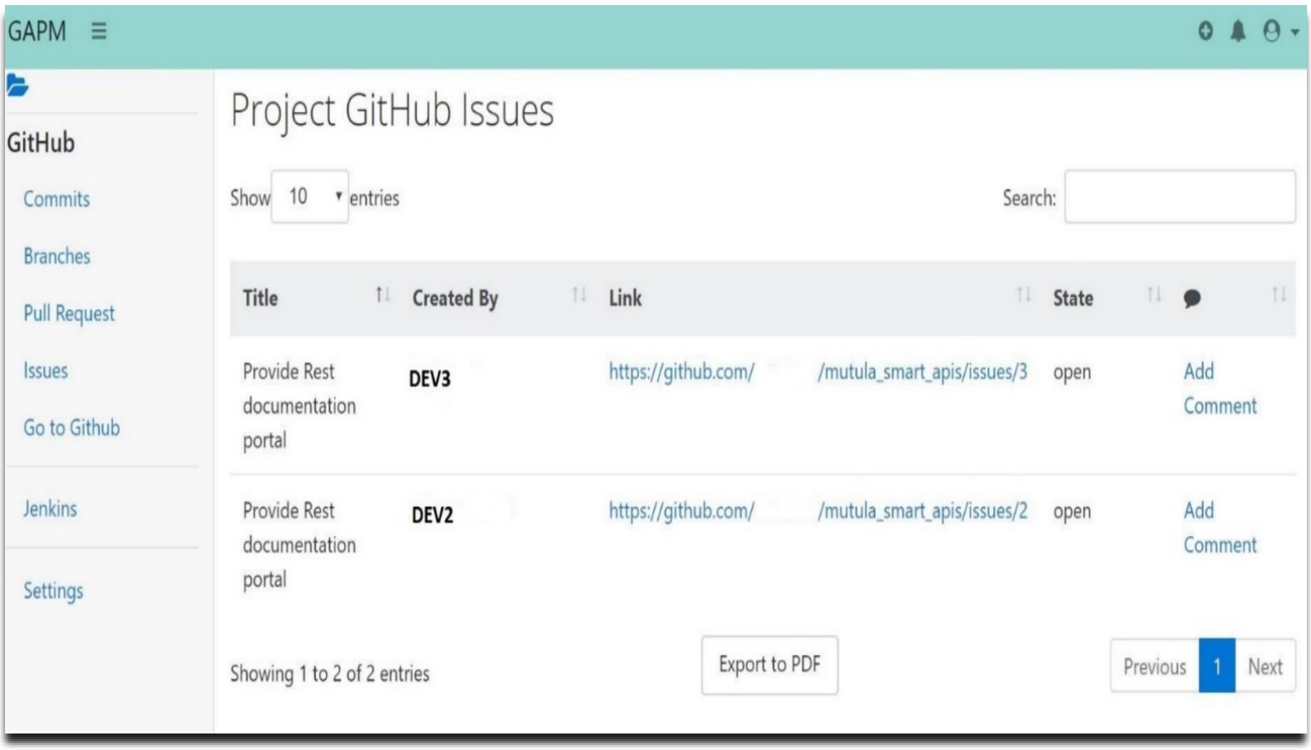


github APP 1:44 PM
[/mutula_smart_apis] Issue created by DEV2 (assigned to DEV3)
#2 Provide Rest documentation portal
Draw a page with clear definitions and functionalities explanation

github APP 3:04 PM
[/mutula_smart_apis] Issue created by DEV3
#3 Provide Rest documentation portal
Once this is done, please ensure test calls are also available on the page

Figure 6.32: ChatOps notification triggered by IS prototype

Both SCM issues were then synchronised and available on the prototype portal to view:



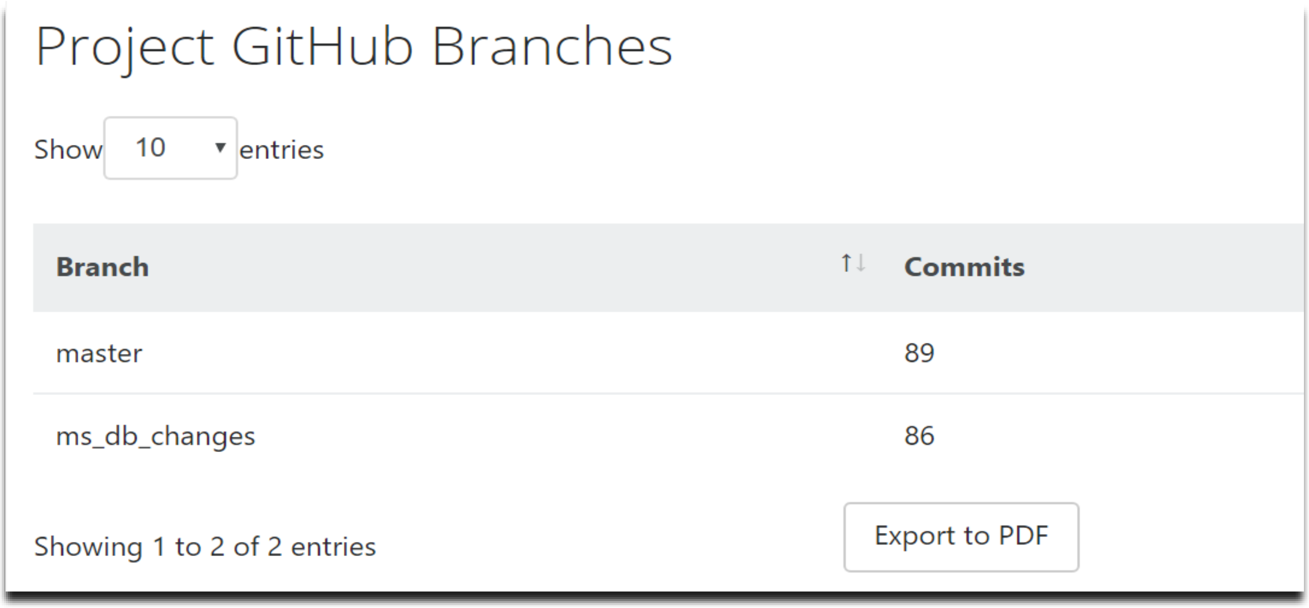
Project GitHub Issues

Title	Created By	Link	State	💬
Provide Rest documentation portal	DEV3	https://github.com/ /mutula_smart_apis/issues/3	open	Add Comment
Provide Rest documentation portal	DEV2	https://github.com/ /mutula_smart_apis/issues/2	open	Add Comment

Showing 1 to 2 of 2 entries

Figure 6.33: SCM created issue and related task generated from the prototype

In other engineering scenarios, the system also provides information on GIT code branches as well as a search facility to reflect SCM changes:

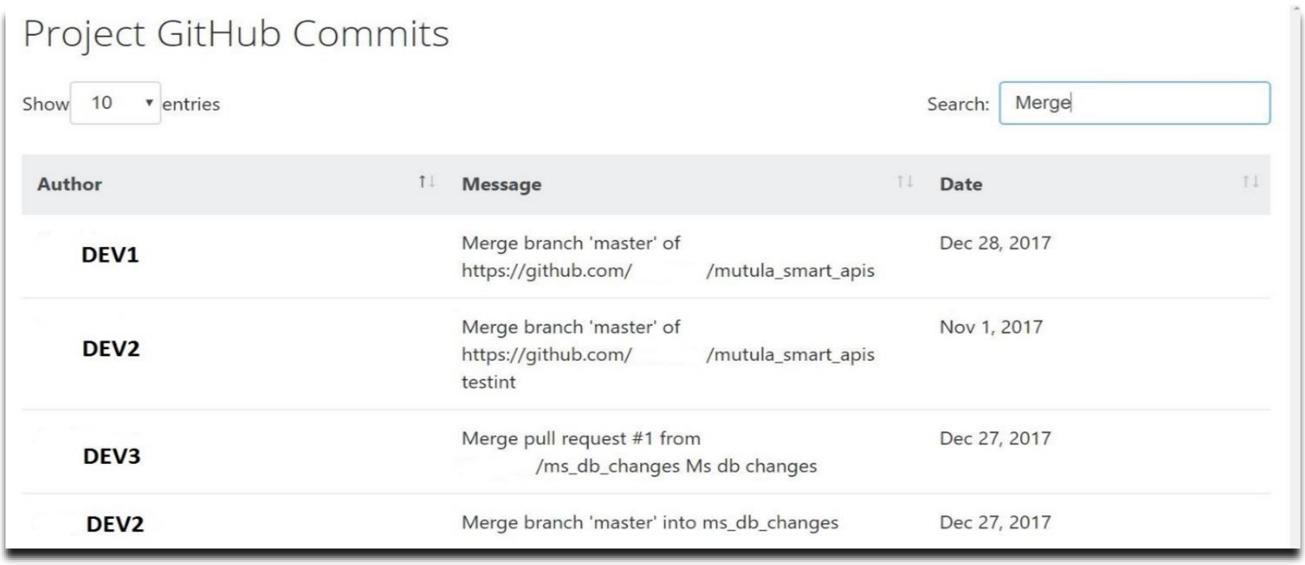

Project GitHub Branches

Show 10 entries

Branch	Commits
master	89
ms_db_changes	86

Showing 1 to 2 of 2 entries

Figure 6.34: Code repository branching data reflected on the IS prototype


Project GitHub Commits

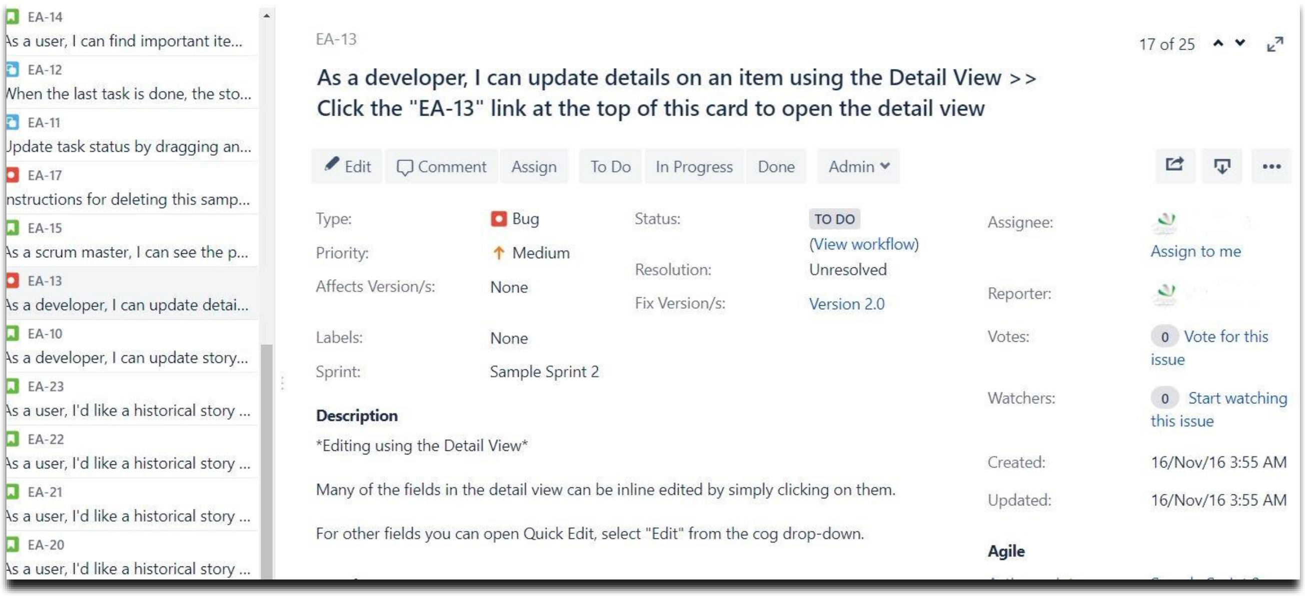
Show 10 entries

Search: Merge

Author	Message	Date
DEV1	Merge branch 'master' of https://github.com/ /mutula_smart_apis	Dec 28, 2017
DEV2	Merge branch 'master' of https://github.com/ /mutula_smart_apis testint	Nov 1, 2017
DEV3	Merge pull request #1 from /ms_db_changes Ms db changes	Dec 27, 2017
DEV2	Merge branch 'master' into ms_db_changes	Dec 27, 2017

Figure 6.35: SCM updates reflected on the IS prototype

Similarly, to SCM related issues retrieved from GITHUB repositories, GZ-AgilePM Consolidator also seamlessly helps project staff view data created, generated and updated in the organisational JIRA project management portal. The next two images illustrate how data which was created on the JIRA portal even prior to the prototype implemented for this thesis is nonetheless synchronised to GZ-AgilePM Consolidator:


EA-14
s a user, I can find important ite...
EA-12
When the last task is done, the sto...
EA-11
Jpdate task status by dragging an...
EA-17
nstructions for deleting this samp...
EA-15
s a scrum master, I can see the p...
EA-13
s a developer, I can update detai...
EA-10
s a developer, I can update story...
EA-23
s a user, I'd like a historical story ...
EA-22
s a user, I'd like a historical story ...
EA-21
s a user, I'd like a historical story ...
EA-20
s a user, I'd like a historical story ...

EA-13

17 of 25

As a developer, I can update details on an item using the Detail View >>
Click the "EA-13" link at the top of this card to open the detail view

Edit | Comment | Assign | To Do | In Progress | Done | Admin

Type:	Bug	Status:	TO DO (View workflow)	Assignee:	[redacted] Assign to me
Priority:	Medium	Resolution:	Unresolved	Reporter:	[redacted]
Affects Version/s:	None	Fix Version/s:	Version 2.0	Votes:	0 Vote for this issue
Labels:	None			Watchers:	0 Start watching this issue
Sprint:	Sample Sprint 2			Created:	16/Nov/16 3:55 AM
				Updated:	16/Nov/16 3:55 AM

Description

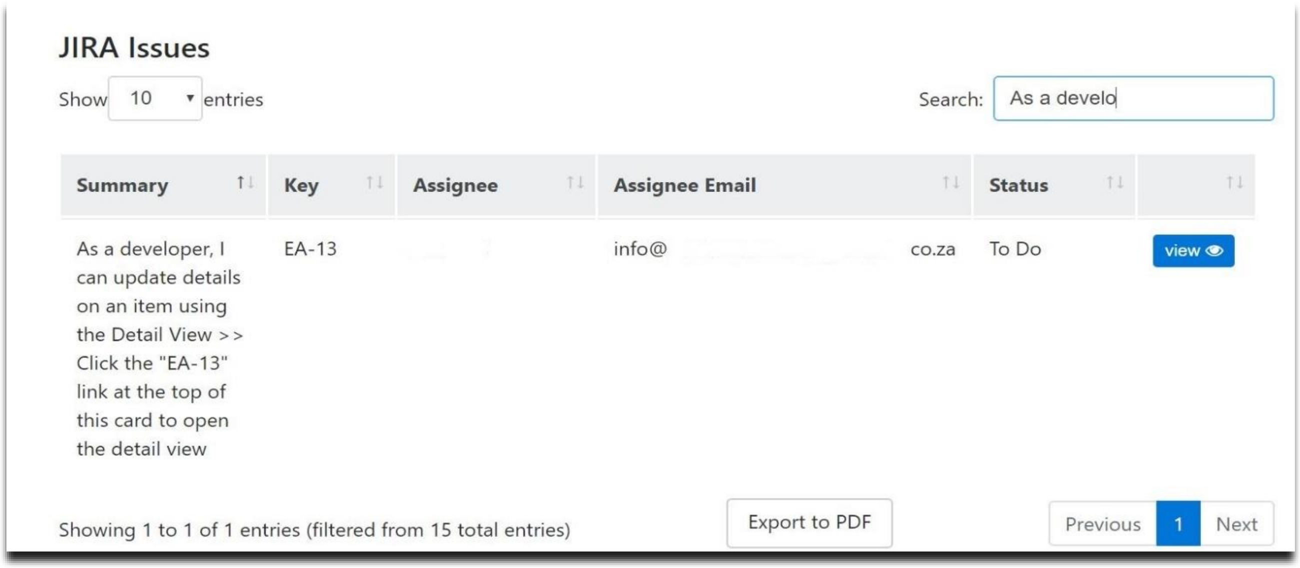
Editing using the Detail View

Many of the fields in the detail view can be inline edited by simply clicking on them.

For other fields you can open Quick Edit, select "Edit" from the cog drop-down.

Agile

Figure 6.36: GZ Consulting Services JIRA issues created in 2016


JIRA Issues

Show 10 entries

Search: As a develo

Summary	Key	Assignee	Assignee Email	Status	
As a developer, I can update details on an item using the Detail View >> Click the "EA-13" link at the top of this card to open the detail view	EA-13	[redacted]	info@[redacted]co.za	To Do	view

Showing 1 to 1 of 1 entries (filtered from 15 total entries)

Export to PDF

Previous 1 Next

Figure 6.37: Data captured prior to implementation as reported by the IS tool

As described during the case study presentation, GZ Consulting Services is an agile engineering shop which advocates for DevOps practices. As such, software as well as operation engineers are expected to work in close cooperation to bridge the traditional gap between these two functions. Consequently, the next observed scenario deals with a simple instance where, following feedback from an operations colleague during regular stand up meetings, software engineering staff members can leverage the IS tool built for the current research to verify build and deployments outcome from CI/CD pipelines. The first step requires that engineers click on the “Jenkins” hyperlink on the left-hand side of the portal:

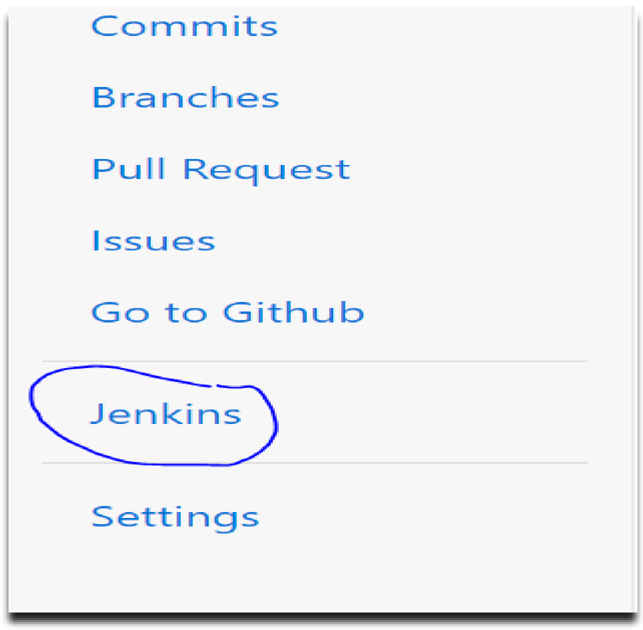
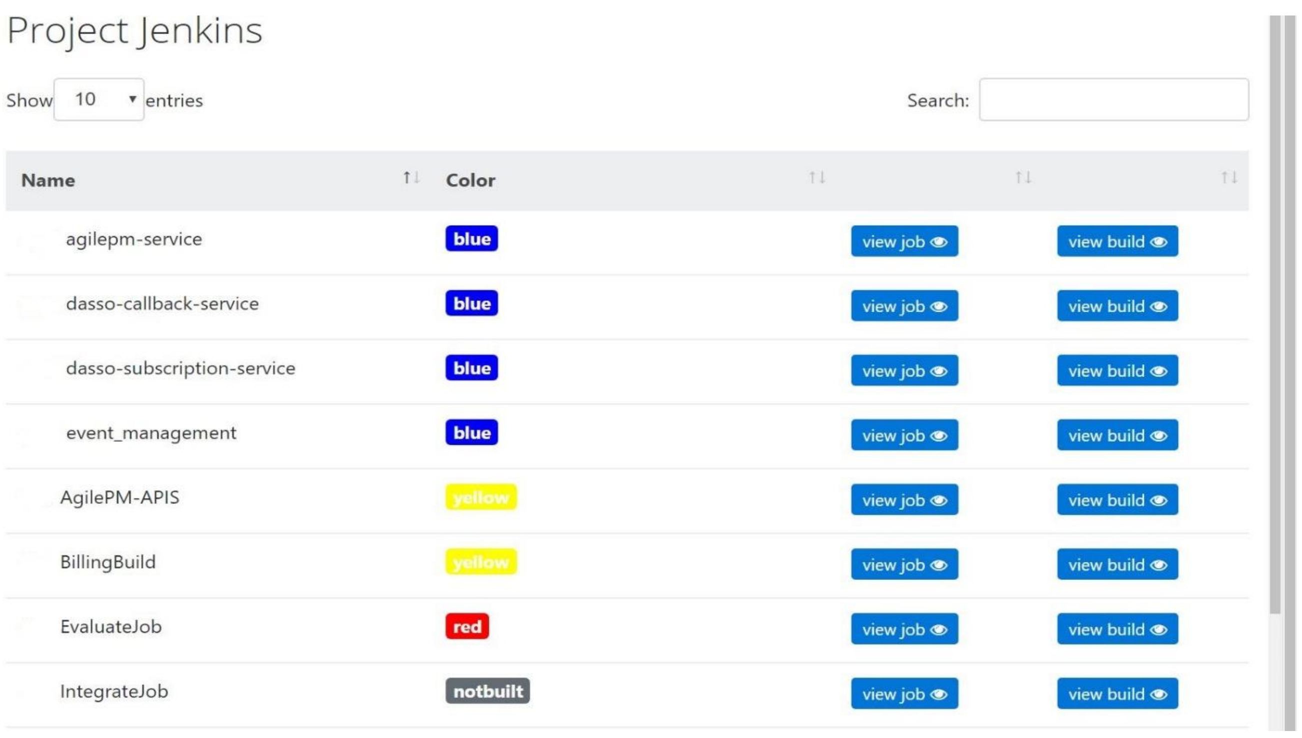


Figure 6.38: Hyperlink to CI/CD data retrieved for the Jenkins organisational instance

The user can now have an overview of all Jenkins related artefacts as well as most recent status (blue for successes, yellow for warnings, red for failures as well as grey for lack of build outcome data).

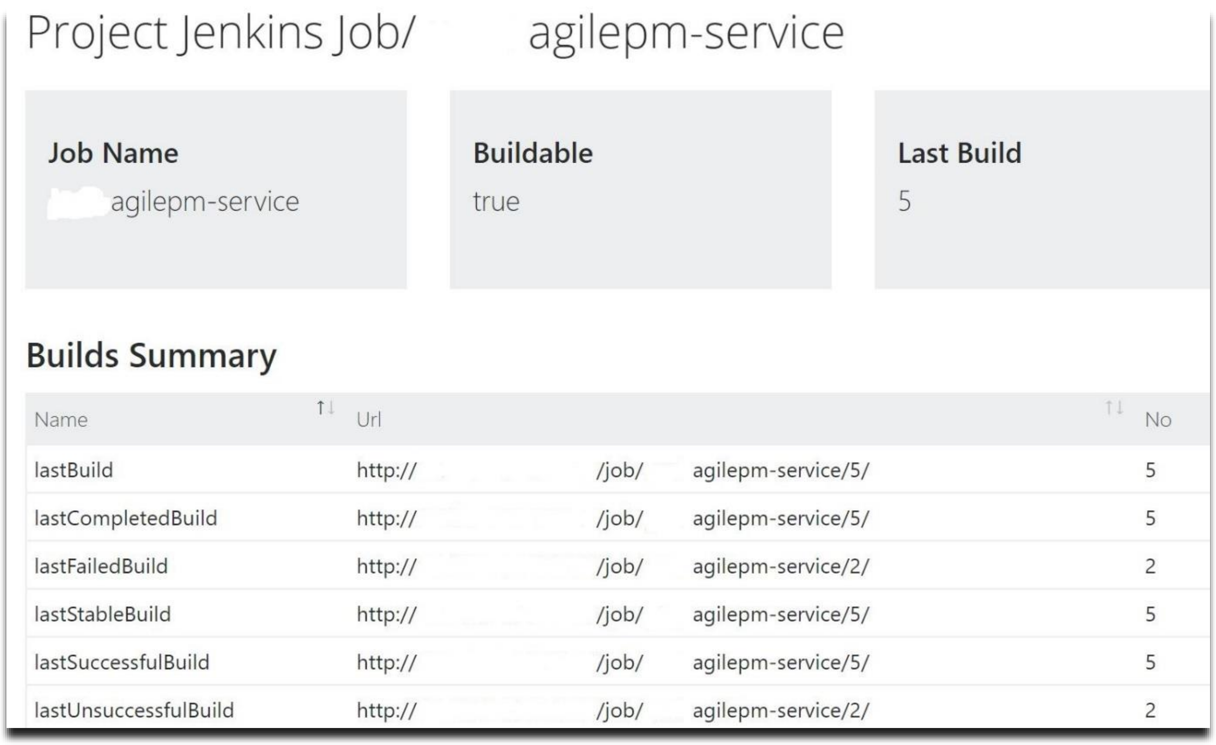

Project Jenkins

Show 10 entries Search:

Name	Color		
agilepm-service	blue	view job	view build
dasso-callback-service	blue	view job	view build
dasso-subscription-service	blue	view job	view build
event_management	blue	view job	view build
AgilePM-APIS	yellow	view job	view build
BillingBuild	yellow	view job	view build
EvaluateJob	red	view job	view build
IntegrateJob	notbuilt	view job	view build

Figure 6.39: Retrieved CI data overview

Users can then drill down further on a specific CI/CD configuration or job for more information:


Project Jenkins Job/ agilepm-service

Job Name	Buildable	Last Build
agilepm-service	true	5

Builds Summary

Name	Url	No
lastBuild	http:// /job/ agilepm-service/5/	5
lastCompletedBuild	http:// /job/ agilepm-service/5/	5
lastFailedBuild	http:// /job/ agilepm-service/2/	2
lastStableBuild	http:// /job/ agilepm-service/5/	5
lastSuccessfulBuild	http:// /job/ agilepm-service/5/	5
lastUnsuccessfulBuild	http:// /job/ agilepm-service/2/	2

Figure 6.40: Build specific data

6.3.6 Post-implementation observational synopsis

From the post-implementation observational process, it emerged that the proposed IS addressed a central issue identified during elaboration of this thesis's research problem: the integration of an agile oriented information system to complement tools already in use at an organisation during engineering projects. The prototype implemented in question was able to deal with the problem of multiple information sources and hence varying versions of truth by aggregating all the data from GZ Consulting Services 'internal agile systems into one consolidated web portal. While the intention of developing such a tool was largely to provide a consistent and uniformed source of truth for all project staff including project leadership as well as engineers, the current research also posited that this consolidation of data would make for better informed projects, and thus improve overall delivery of software project quality.

To achieve a better understanding of this situation, the researcher opted as previously indicated to distribute an electronic survey to study participants. This Activity Theory themed questionnaire aimed to get views of management and engineering staff about quality software delivery and agile development. The next section proceeds to a discussion on data and insight retrieved from this electronic survey.

6.4 Analysis of data collected through the electronic survey

To gain insight into the opinions, perceptions and experiences of GZ Consulting Services staff, an Activity theory themed electronic survey was designed and implemented through the Limesurvey platform. The survey was described to participants as an essential component of an exploratory study in quality software delivery as well as agile development in a sub-Saharan start-up context. The questionnaire constituted an attempt to explore participants' perceptions of quality software delivery in their day to day functions prior and post the implementation of this information system. The survey was centred on the following main themes:

- Software quality and agile philosophy practical applications.
- Employee user experience with GZ Agile Project Management Consolidator (the experimental IS tool).
- Employee opinion of GZ Agile Project Management Consolidator and its perceived impact on engineering projects performance and quality improvement at the firm.

Figure 6.41 describes participant functional areas surveyed:

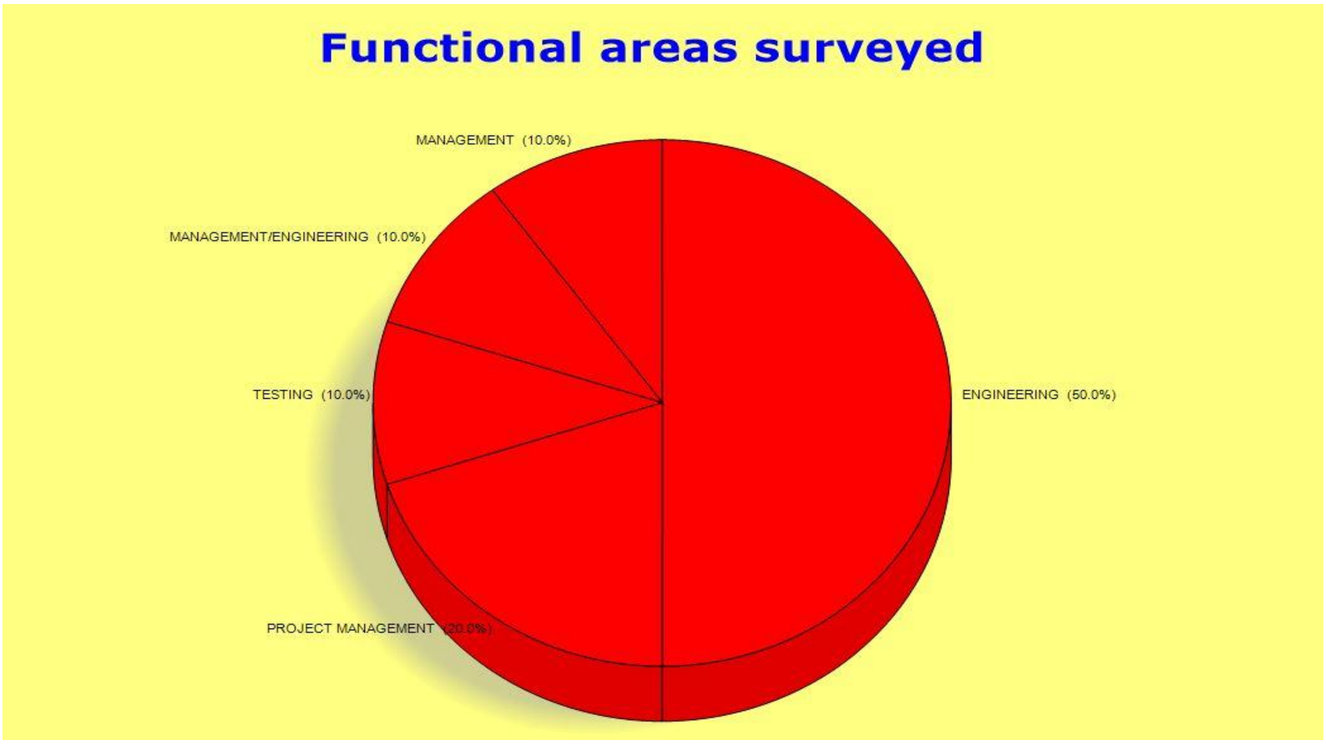


Figure 6.41: QDA Miner surveyed functional areas

Half the participants surveyed (50%) were technical staff whose function was to essentially conduct engineering related activities. These activities included software development as well as DevOps cloud infrastructure related scripting and deployment tasks. The other half of staff which participated in our survey included project managers (20% of respondents), testing specialists (10%) as well as senior leadership (20%) which was in odd cases additionally involved in some capacity in certain engineering adjacent activities. Table 6-12 illustrates a sample of survey participants' demographic profiles:

Table 6-12: Sample participant profile

JOB POSITION	**GENDER (M/F)**	**QUALIFICATIONS**	**AREA**	**EXPERIENCE (YEARS)**
Engineer	Male	Bachelor's Degree IT	Software Projects	1-5
Engineer	Male	Bachelor's Degree IT	Software Projects	1-5
Engineer	Male	Bachelor's Degree IT	Cloud Infrastructure	1-5

Engineer	Male	DIPLOMA IT	Software Projects	1-5
Engineer	Male	Bachelor’s Degree IT	Software Projects	1-5
Project Manager	Male	Master’s Degree Engineering	Software Projects	5-10
Project Manager	Male	Masters Business	Software Projects	> 10
Tester	Male	DIPLOMA IT	Software Projects	1-5
Manager	Male	Master’s Degree IT	R&D	5-10
Manager	Male	Master’s Business	Executive	>10

Table 6-12 presents of staff which participated in the electronic survey for the current study. These individuals are the subjects which represent actors within a community in the proposed conceptual framework. Participants were selected from various functional units of GZ Consulting Services. They were picked by matching their day to day responsibilities with the main activities of the Activity Theory conceptual framework which revolves around engineering and project management activities. In subsequent sections, Chapter Six proceeds to discuss in further detail insight retrieved from the subthemes which informed this electronic survey.

6.4.1 Staff perceptions on quality and agile philosophy practical applications

To get a sense of how GZ Consulting Services perceived the notions of quality and the agile philosophy in software engineering projects context, they were first asked to describe these terms in their own words. Table 6-13 illustrates a sample of GZCS staff responses during the electronic survey regarding the concept of quality in engineering contexts as well as the agile approach in general.

Table 6-13 Sample participant feedback on quality and agile

SURVEY QUESTION SNAPSHOT	PARTICIPANTS FEEDBACK

What does the term quality mean to you as an employee?	1. *"Getting the very best out of something." (Participant 1)* 2. *"It means providing quality software that is reliable and provides accurate data at all times" (Participant 2)* 3. *"Quality means that all errors are eliminated (...) (Participant 3)* 4. *"Delivering clean, reusable and scalable code that complies with best practices"* (Participant 4) 5. *"(..) The standard of a product being delivered" (Participant 5)*
Please describe in your words Agile in software engineering contexts	1. *"(..) Ability to adapt (...)" (Participant 1)* 2. *"It is an iterative, incremental method of designing software as opposed to waterfall approach. Development team must be able to be flexible and react quickly to requirements changes during software development" (Participant 2)* 3. *"It helps creating an improved work rate and see how effective we work as a team on a particular project"* (Participant 3) 4. *"(...) In short, the philosophy is to make constant delivery as smooth as possible. (Participant 4)* 5. *"(...) An incremental way to develop software(...) (Participant 5)*

In the introductory line of questioning, it appeared that staff had an informed grasp of what quality and agility entailed in software engineering contexts at GZ Consulting Services. When referring to the concept of quality, participants used key phrases such as *the elimination of errors*, terms including standardisation or norms regarding development processes while stressing the need to consistently deliver reliable products.

When asked about their understanding of the agile philosophy and related methodologies, staff members who participated to the survey constantly referred to agile terminology in mostly software engineering as well as modern agile project management jargon. Some of the terms that were often cited by leadership and engineering staff which participated to the electronic survey included for instance *incremental and iterative* approaches to software development on one hand; and on the other hand, the improvement of productivity by putting an emphasis on the ability for modern engineering team to adapt to evolving requirements and requests from customers.

Next, participants were queried on their view about the facilitation of quality software delivery by leveraging agile development processes as described as described below:

Table 6-14 Snapshot of perspectives on agile practices and their impact on quality delivery

SURVEY QUESTION SNAPSHOT	PARTICIPANTS FEEDBACK
In this study, it is argued that agile development processes facilitate the delivery of quality to markets. Does this assertion seem accurate to you? Please explain why.	1. *"(..) Having requirements weekly from various clients can be quite cumbersome (..) Agility helps us anticipate changes required, provides flexibility because of the environment's dynamic nature, and as changes are made in each iteration, as a result quality is provided without wasting time." (Participant 1)* 2. *"Agreed. Agile processes focus more on objectives to be met than the processes in themselves. You get a sense of having achieved some level of user satisfaction before having actually deployed the software in my view." (Participant 2)* 3. *"Yes, I agree with the statement because by making the development process more efficient, software quality of the product is improved as our experience at GZCS indicates to date. (Participant 3)* 4. *"(…) I have myself experienced this improvement as improving the development process has improved the delivery process and reviewing areas of the application to be delivered which need improvements." (Participant 4)*

	5. *"(..) The basic idea is to find better ways of building software more simply. (...) Agility eliminates a huge part of risks associated with delivery of large projects by providing ways to work in an iterative and incremental manner which also allows the team to gather feedback proactively and react quickly" (Participant 5)* 6. *Not exactly, while I believe agile is helpful, I think its main contribution is in improve delivery of software to markets and not necessarily guarantees quality. (Participant 6)* 7. *(...) This assertion does seem accurate to me because it is a well-known and proven approach to high quality software delivery. Over time, the approach of starting development and improving or reconsidering certain aspects of the software at some point of the development process has proved to be working well and avoid time wasted defining extremely detailed and exhaustive specifications. Faster delivery timeline is what drives the Agile development process in my view. (Participant 7)* 8. *The assertion seems accurate in the sense that agile processes enable the team to deliver smaller working versions of the software to the users, thus collecting feedback from the real users very early in the process. The feedback from users is used as input early in process to improve quality of the product. The iterative nature of such an agile approach and the consultative process used tend to increase the perceived quality of the software. (Participant 8)* 9. *Yes, because I think we pick up bugs quicker and deliver solutions faster. (Participant 9)*

From the discussion on agile processes implementations impact on the delivery of quality software products to markets during the electronic survey, most employees who participated seem to believe in correlation between these two aspects.

When looking at participants answer in the detail though, it seems that most subjects believe that agile practices improve the delivery of products to market. Participants who agree with this view also suggests that because of the constant involvement of users during the development processes, engineers are better equipped with identifying problems early and rectifying them thanks to the adaptive nature of agile processes. This they argue, reduces defects, improves anticipation to evolving needs and thus improves the delivery of quality products.

One respondent nonetheless advanced a different view from most stuff with whom the researcher had interactions during data collection. His views presented an argument which although did highlight the most shared sentiment amongst staff, cannot be dismissed as it does not completely appear as unfounded. This staff member mentioned that in principle, he agreed that agility at large through the adoption of agile methodologies and practices at GZ Consulting Services had a positive impact in delivering products to markets. In his view, this has obviously proven to have a huge positive impact at GZ Consulting Services. Previous research has also suggested that organisations which adopt such agile methodologies, approaches and tools will often notice significant improvement for their businesses if one considers all the issues which have traditionally plagued on time delivery of software projects (Jones 2010, p. 7). Yet, the participant mentioned earlier in question differed with his colleagues in their appreciation of agile adoption and its impact on quality. To further explain his point, this participant to the study claimed that he did not see a direct correlation between improvements in delivery timelines thanks to agile practices, and actual quality benefits in the product itself. This is a sensible view if one requires quantitative metrics on software artefacts themselves. However, recent research has nonetheless also shown certain limitations when there is an overreliance of quantitative metrics to evaluate project success especially in terms of software quality. For example, although some have traditionally argued for the consideration of Source Lines Of Code (SLOC) as a foundational variable in quality improvements endeavours, it is actually weakly correlated to so many other factors that it cannot in itself be considered as an accurate indication of quality (Voas & Kuhn 2017, p. 21). As such, the benefit of qualitative studies in engineering context includes a need to consider the interpretative nature of phenomena; and without necessarily ignoring quantitative data, also cater for other variables such as user perceptions, biases, internal as well as organisational contexts etc. In this light, the next line of questioning attempted to gather views of employees about GZCS staff experiences in software projects prior to the implementation of the current researcher's prototype during the electronic survey.

Table 6-15 Electronic survey responses summary about their personal software project experiences at GZCS

SURVEY QUESTION SNAPSHOT	PARTICIPANTS FEEDBACK
A new engineer is onboarded to perform pair programming duties on a software projects you have been leading to date. Where do you direct him to get all the information he or she requires?	1. *As for documentation, I would refer the engineer to our JIRA profile and for technical information such as code, files etc. I would refer him/her to our GitHub repository.* 2. *We direct them to Confluence which is a document management tool where all our documentation is stored.* 3. *Via links to the platform's appropriate pages provided in an email.* 4. *I will direct the new colleague to the project git repository.* 5. *To the agile management system JIRA* 6. *(…) The GIT archive (…). This will make the adaptation process easier and quicker.*
Prior implementation of GZ Agile PM Consolidator, how did you typically obtain descriptive and operational information about software projects at your firm?	1. *Usually via mail or getting it from a repository.* 2. *From meeting with clients (...)* 3. *Information was obtained mainly via email.* 4. *(…) Documents shared on Google docs and Dropbox. We often had briefing sessions in which we went through requirements and tasks were managed through the issues queue on GITHUB.* 5. *Mostly using various isolated tools*

	6. *By engaging with different stakeholders separately.*

From the line of questioning discussed above, it emerged that existing staff almost exclusively relied on the wide array of various information systems used at GZCS to get information required to inform their decision-making processes in order to perform their daily tasks. For engineers, the common thread seemed to be a reliance on GITHUB, and JIRA and these two tools would typically be their point of references when onboarding and guiding new colleagues.

Having obtained a feel for what the ambiance or processes in use at GZCS prior to the current research, the study proceeded to gather staff perceptions post the implementation of the proposed experimental information system.

6.4.2 Employee user experience with the implemented experimental tool

Feedback retrieved after analysis of staff feedback from the electronic suggested that more than half of employees at GZ CS had experienced some sort of interaction with the proposed IS. The pie chart below illustrated participants' feedback when queried on their usage habits:

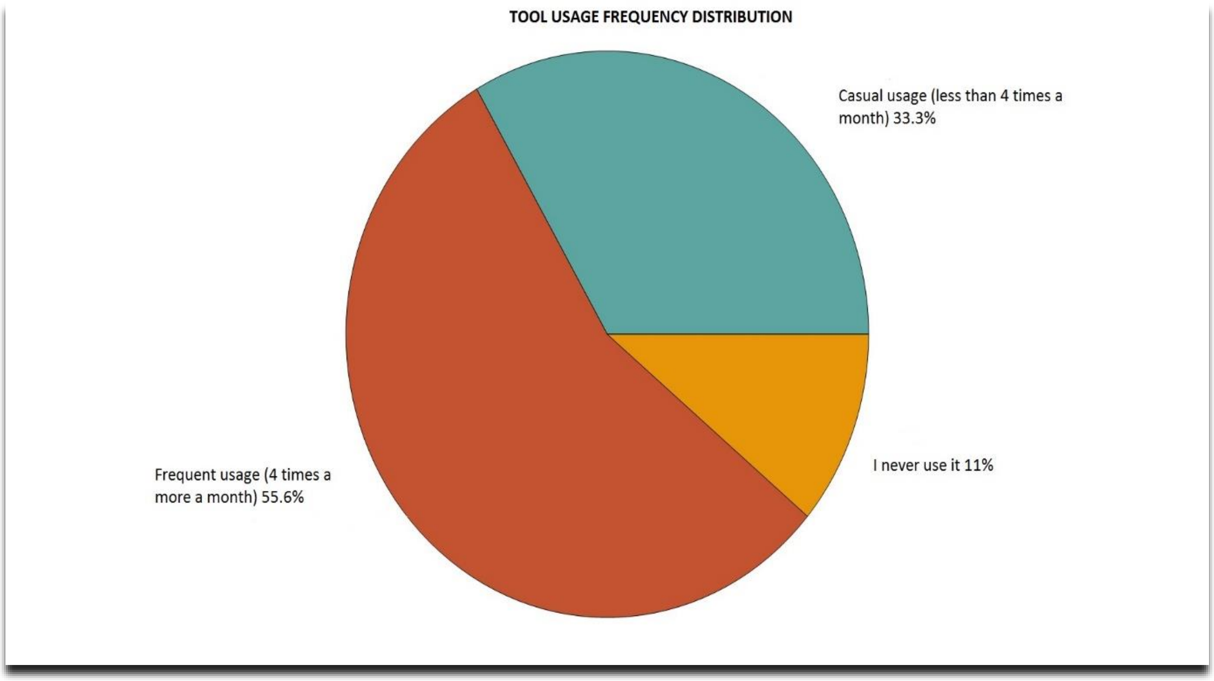


Figure 6.42: GZ-AgilePM Consolidator usage frequency distribution

Figure 6.42 seemed to indicate that most staff who participated in this study described their usage of the tool as frequent. In the next tables, data gathered about employees' experiences with the prototype follows:

Table 6-16 Summary of staff user experience with the experimental tool

SURVEY QUESTION SNAPSHOT	PARTICIPANTS FEEDBACK
How would you describe your first impression of the prototype deployed at your firm during this research?	1. *(...) I would say it is a great platform to keep track of all your projects and team members(...)* 2. *It empowers the delivery team with a consolidated view across all siloed commercial Agile tools.* 3. *Reporting and enabling tool for software development and project management in agile environments.*
Please describe the ways in which you use the tool	1. *I use the system on the go, meaning I keep track of my team and projects assigned to me.* 2. *(...) Managing software projects and related git, Jira and Jenkins artefacts via a single user interface. It saves a lot of time during development.* 3. *Reporting and information clarity*
What incentives do you have to use the system?	1. *I would say time saving. I am just a few clicks away from getting the tasks assigned to me* 2. *Better development time, time saving and high throughput from the development team.* 3. *I don't have to logon on Jira and GitHub and Jenkins all the time.*

Would you say it was easy to get used to this system? Please explain	1. *Yes, because from my perspective as a developer, navigating was straight forward.* 2. *Yes, the system is friendly and very convenient to the developers* 3. *Yes because of a simple and intuitive interface.*

An overall majority of respondents seemed to have a favourable opinion regarding the experimental tool proposed at GZCS for the current research. Employee's first impressions on the IS after its introduction at the firm included a perceived usefulness on the reporting of agile related project information thanks to the consolidated portal. The initial assessment seemed to be that they were happy with reporting aspects of the system.

One of the recurring positive aspects also mentioned by users is that they appreciated the user-friendly interface. This is because the learning curve to get used to the system is not deemed to be steep at all for technical or project management personnel alike. When asked about what their key motivations outside of contributing to the current study where in utilizing the tool, users typically responded that as they got used to the system, they found that it was saving them a lot of time in their day to day tasks. It appeared to be the case for technical and non-technical staff alike. This, in respondent's eyes was beneficial as the portal provides a unified view of information about code source control metrics, CI artefacts deployment status, as well as issues tracking data aggregated from all the systems in use at GZ Consulting services for the management of agile software projects. In the next section, a summarised version of interrogations put forth to participants concerning perceived quality and performance related gains due to the tool's implementation are further discussed.

6.4.3 Tool evaluation and its perceived impact on engineering projects at GZCS

Concerning their perspectives in evaluating whether the implementation of GZ Agile PM Consolidator had a positive impact at the firm or not, most staffed appeared to view the system favourably. This is because according to them, its usage proved to be simple and intuitive for most users. Moreover, a key perceived benefit was that it provided effective integration between the large array of tools used at GZ Consulting Services for the management of software projects in a very intuitive fashion.

As mentioned earlier, most of the system's end users appreciated the user friendliness of the tool as well as its approach to information aggregation and presentation to relevant stakeholders in agile and DevOps contexts. Yet, as no system can be perfect, there was an additional need to synthesise what users felt about the portal's impact on their work. Additionally, to arrive at a more succinct understanding on employee's views concerning possible improvements that the system could benefit from, respondents were asked to put forth their views and suggestions to the researcher.

The themed question presented to them in this case could be summarized as follows: "*What in your view is the systems impact on quality and performance at your firm*? *Please try and elaborate this impact be it positive or negative...*". Subsequently, respondents were also asked to provide suggestions for areas of improved if possible. In the following sections, a synthesis of retrieved feedback from users of the system follows.

Table 6-17 Respondents perspectives on performance and project quality improvements

- *"The system does contribute to improve developer's performance in my view. As explained earlier, it saves me a lot of time, so I can focus on engineering specific activities as per my actual responsibilities. I think it also improves overall work quality because since tasks are assigned per individuals in a team, typically, everyone must first make use of JIRA, and then go back to GitHub and try guess at times which items are related or correspond from each platform to keep track of the items they must work on. With GZ AgilePM consolidator, the team has a more intuitive way to keep track of items in a consolidated manner and this yields a lot less confusion then what we have be accustomed to in the past." (Participant 1)*
- *"Yes, the system improves performance because instead of having siloed views, we now hence a precise overall outlook on project progress. It improves work quality by making our development process more efficient and this has helped us produce better quality products in my view". (Participant 2)*
- *"The new portal improves our product quality in my view from project leadership view point. It has for instance improved the accuracy of my reporting to stakeholders by not only enhancing the collaboration state of mind between engineers, but it has also helped me better manage the team and work allocations for example. This has had a direct impact on my relationship with business and end users of our software because we have been better prepared to anticipate client needs and hence make*

necessary adjustments before certain bugs are even picked up by clients". (Participant 3)

- "*A typical part of my day has typically revolved around accessing GitHub, JIRA as well as Jenkins separately to perform tasks a DevOps engineer is supposed to attend at the firm. I have seen an increase in my productivity since the tool came in usage because most of these tasks are integrated through the AgilePM Consolidator. So, for instance where I would typically spend 2 to 3 hours trying to understand a task request on JIRA, send comments or emails when there was a lack of clarity on the board, wait for related task to be created on the GIT system etc. I can now seamlessly have access to that information and send communication via one location. I can also see updated Continuous Integration builds and deployment information through this portal without having to navigate to each front on all our Jenkins clusters. (Participant 4)*
- *"Yes, it enhances our collaboration in a ubiquitous manner and helps deliver by providing clarity in the team, so everyone is on the same page." (Participant 5)*
- *"The Agile Consolidator improves the fast tracking of development activities with its form of integration of functionalities from external back end systems into one suite. We have already seen improvements in terms of onboarding new team members, they are up to speed in virtually no time where we would typically wait almost 3 to 5months before new staff are fully operational and versed in our technology stack." (Participant 6)*
- *"Personally, I feel more confident about my role as a project manager tasked with leading scrum sessions and provide overall project leadership. The system has enabled me to receive proper feedback and be able to reconcile this information with often informal information flows received via emails or ad-hoc conversations. The system definitely has improved delivery of quality projects under my watch because the monitoring and auditing aspects of the reporting helps eliminate confusion among team members which creates a better structured environment for all." (Participant 7)*
- *"The tool improves performance of the team in the sense where it provides a single platform to perform all engineering and project related activities. This has improved project quality by introducing a level of standardisation across all underlying tools at GZCS. It has also raised in my opinion the level of motivation for the team thanks*

to the transparency it has fostered, now everyone receives the information in real-time and can contribute in a simpler manner to all projects." (Participant 8)

Table 6-18 Respondents suggested improvements for GZAPMC

- *The UX is already there, but I think the UI need more work, and the report could be not improved but enriched. Like pulling report per assignee or priority level. Also, ability to draw contrast from the data we can access would be cool. (Participant 1)*
- *Include a chatting module to allow team members to communicate (a bit like slack). (Participant 2)*
- *So far so good (Participant 3)*
- *Keep more pressure on us and make sure we deliver our work on time (Participant 4)*
- *There are a few bugs here and there but besides that, I believe that it is ready to hit the market and perform as the tool it is supposed to be. More integration of tools would be advantageous, but it will have to be studied before proceeding since the market is flooded with all sort of development tools these days. (Participant 5)*
- *The solution should enable more data visualization around the use of various integrates platform (e.g. GitHub, Jira...) to have a better view that will create easy report that can be understand by anyone. (Participant 6)*
- *A native integration with AI and chat bot would be more interesting, this will ensure that some projects are challenged by the AI before they are created to avoid potential project duplication. (Participant 7)*
- *Add more integration to the actual cloud infrastructure (Participant 8)*

Most participants seemed to indicate their satisfaction with the proposed information system regarding its initially stated objectives. The prototype was considered by staff to be a time saver as it helped employees focus on login on to one central portal and being able to execute most of their agile related functions with this interface. Survey respondents indicated for instance that by providing a single platform for all project activities, there were significant time gains by project members and an enforced sense of standardization.

Agile project leadership including management reported on their side that they felt there was an improvement on overall quality delivered to their customers and key stakeholders. This they argued, was simply because this new tool had for instance help improved their reporting by providing a more accurate and unified view of information. Nonetheless there were some suggested improvements in terms of increasing the data sets available for all the tools the system integrates, bug fixes, instant messaging and chat bot improved functionalities as well as beautification of the front end among others.

6.5 Chapter Six Summary

This section of the thesis proceeded with a discussion on research findings by first indicating how anchor codes where generated from the research questions established in Chapter One.

Next, initial findings were introduced and followed by a description of the inductive coding and subsequent themeing used for data findings. A qualitative data analysis of semi-structured interviews transcripts was then executed with an emphasis put on content and frequency analysis to derive further insight from the interactions with participants. Interview subjects were fluent in agile jargon and practices. They also indicated that while agile was effective at GZ Consulting Services, there was a need to not only remain customer focussed, but to also put an emphasis on improving internal communication while tweaking internal processes to ensure agile projects yielded quality solutions to clients.

Activity Theory was used as the central theoretical lens in the case of both the observational process and distributed electronic qualitative questionnaire. The central activities underpinning this questionnaire revolved around agile project management and engineering be it pure code development and DevOps functions. From the analysis of data gathered during these processes, it emerged that users of the systems were mostly satisfied with the tool proposed and the main suggested improvements were in terms of added functionalities, bug fixes and front-end improvements. In the latter parts of this thesis, a final exercise in interpreting research findings with the objective of proposing a new AT inspired theoretical framework follows.

7 CHAPTER SEVEN: INTERPRETATION OF FINDINGS

In the previous chapter, findings retrieved were presented using research procedures described in the methodology portion of this inquiry. Data analysis was then conducted with the aim of understanding how GZ Consulting Services as an organisation went about delivering quality solutions to markets through the leveraging of agile methodologies and technology tooling in the execution of its software projects. Chapter Seven serves as the conduit for a discussion on interpretations and insights gathered from findings exposed during this research.

7.1 Study context

The researcher looked at GZ Consulting Services from the perspective of its engineering and agile project management activities. This was done both prior and post implementation of a prototype information system to address stated research objectives in Chapter One.

To further gage whether the introduced IS prototype addressed research objectives, one could on the first hand proceed to draw from Activity Theory. AT was selected as an underpinning theoretical framework during the discussion on research methodology. This was done to further elaborate on the activity system in place used for the delivery of quality software engineering projects and solutions at GZ Consulting services.

Analysis was first done to explore the situation at GZCS prior to implementation of the current study's proposed IS prototype. Consequently, study subjects and the tools at their disposable to arrive at a specific set of given objectives in their daily activities are illustrated in the following diagram:

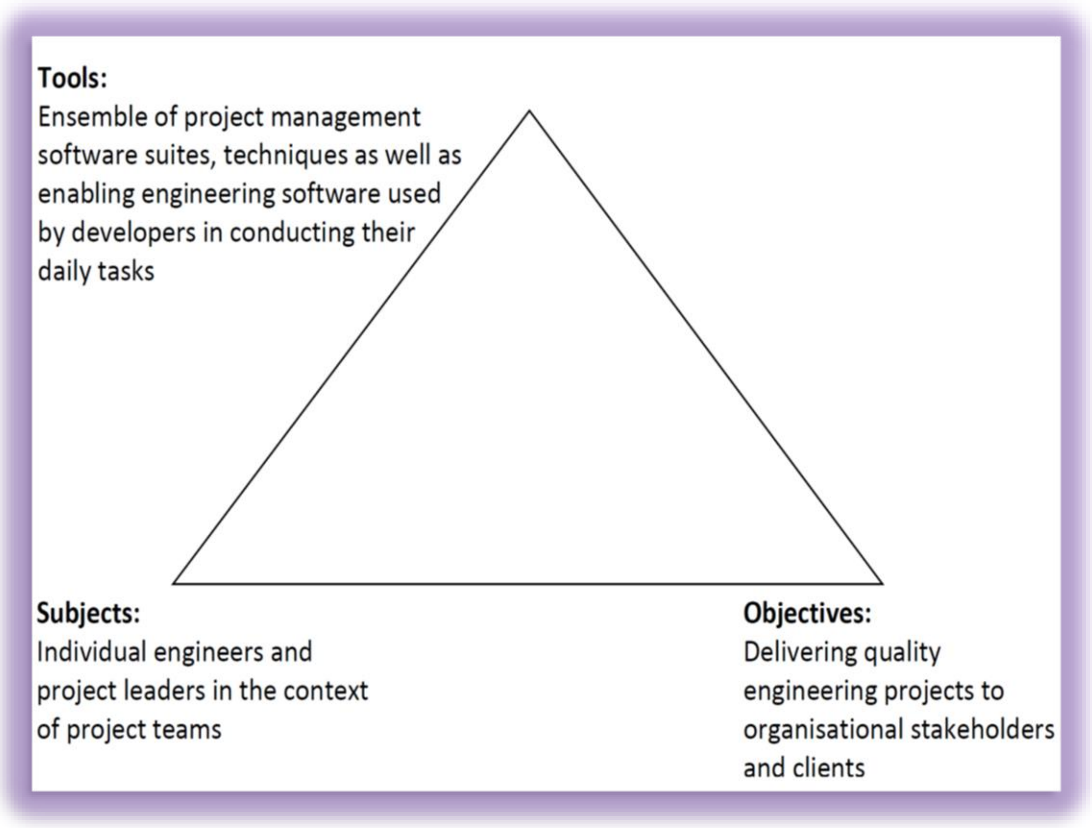

Tools:
Ensemble of project management software suites, techniques as well as enabling engineering software used by developers in conducting their daily tasks
Subjects:
Individual engineers and project leaders in the context of project teams
Objectives:
Delivering quality engineering projects to organisational stakeholders and clients

Figure 7.1 Basic AT perspective of situation at GZCS prior to current study

What was illustrated from insight gathered before implementation of this study's IS prototype is an activity system which revolved around project team members, their objective of delivering quality solutions to stakeholders, and their leveraging of various project management software suites, engineering techniques as well as related tooling to achieve these objectives. This activity system essentially comprised of the activities executed by observed GZCS staff in their daily tasks. The performing of these tasks was informed by a need to execute engineering software projects in such a way as to deliver quality successful outcomes for project teams as well as customers. In engineering projects contexts, this was facilitated by leveraging relevant agile methodologies, techniques and appropriate technology tooling. Yet, these objectives were not attainable without challenges, and initial observations seemed to attest to this.

In effect, participants to the study exposed that some issues were impeding their ability to produce and deliver quality solutions in an optimal fashion. For example, project managers advanced that GZ Consulting Services was effectively using the best of breed methods and technological tools in building its software products and solutions. Yet, it was suspected that team members were using so many different software products to conduct their day to day tasks in the context of engineering projects and normal operations; that this situation was in fact encouraging staff to work in silos. Although DevOps approaches have been advocated by industry players to deal with issues related to siloed workstreams between development and operational functions, some challenges persist (Masombuka & Mnkandla 2018, p. 279).

At GZCS for instance, poorly managed agile technology usage created a situation where each member was only interested in the tools required for his or hers function and did not leverage anything outside their perceived scope of required responsibilities to better inform their decision making. From this, resulted a side effect in the form of large arrays of information views generated by the ensemble of software packages utilised by project teams. Consequently, decision making was often ill advised because the leadership in both project management and engineering functions did not have sufficiently contextualised information. This created a very problematic situation for project leadership as well as technical team leads who needed to account for project lifecycles and ensure that only the best quality artefacts were delivered in a required manner for stakeholders. The need to consolidate information views for project teams and hence help them improve their chances at delivering quality software projects motivated the development of this study's proposed IS solution.

From a Vygostkian perspective, it is additionally important to understand human activities as collective rather than individual within most social and organisational contexts (Hardman 2005, p. 381). For this study, activities which were collectively executed by staff with the aim of delivering quality software engineering projects as well as products to stakeholders and clients were hence identified and explored. The next section proposes an interpretive lens on this study's findings.

7.2 Interpretations

7.2.1 Exploring initial research findings

Qualitative questionnaires typically involve submitting series of predetermined questions in order to obtain feedback from participants. These interrogations are usually designed in a manner which researchers deem relevant to the inquiries at hand. Some of the key advantages provided by these tools include their cost effectiveness on one hand, but mostly their ability to enable insightful observation of phenomena as well as the analysis of processes in social and organisational contexts (McQuirk & O'Neill in Iain 2013, p. 191).

Using questionnaires as a data collection tool in this study was considered useful in understanding how the proposed IS prototype was being utilised and appreciated by GZCS staff. The researcher also sought to find ways and means in which the portal could be better adapted in order to further assist engineering and project management staff in their daily activities aimed at delivering quality solutions.

The following sub-sections further explore and interpret research findings regarding usage of the proposed IS tool through an overall evaluation of the situation at GZ Consulting services as the current study' experimental component concluded.

7.2.2 Insight on quality delivery and agile at GZCS

During initial conversations about quality in engineering contexts, GZ Consulting Services staff conceptualized as a central key requirement the need to standardize processes. Although other aspects such as the maximal elimination and management of software errors were also evoked, emphasis was put on the need to have norms in terms of processes. This fit in with a conversation between the researcher and participants about the agile philosophy and related methodologies. These were indicated to enhance consistency and predictability in the delivery of quality agile software projects for organizations and their customers.

Most participants to the current study described quality from the perspective of their daily responsibilities at GZ Consulting Services. What seemed to appear was that leadership and project management staff mostly described quality in what could be considered common usage jargon. Some of these employees for example described quality in phrases such as "*getting the best out of something*" or "*the standard of a product being delivered*". On the other hand, engineering staff used a lexicon which can be described as more software development specific in referring to quality. Staff members familiar with agile precepts associated both well managed engineering processes as well as agile practices with quality.

As presented during exposition of research findings, the bulk of GZCS staff which participated to this study viewed agile practices in a favourable light. This did not seem at first glance to be directly correlated with their overall seniority level in the software engineering field; but rather, with the amount of experience they had in practical agile settings. Staff members perceiving agile approaches to their work in a favourable light is very significant in increasing their enthusiasm and affects delivery of quality outputs in the context of agile software engineering projects. This postulate is aligned with studies which have for instance shown that, increased team motivation for engineering team is critical in improving quality management, productivity and ultimately successful project delivery (Beeccham et al. in Steinberga & Smite 2011, p. 117).

7.3 Validating the AT model outcome

7.3.1 Insight on GZ AgilePM Consolidator

Some research has suggested that organisations need to not only facilitate self-organisation of their engineering teams, but that these teams need to additionally adjust their behaviour and tune their tools accordingly to better improve their chance at delivering quality projects (Kalermo & Rissanen 2002, p. 152). In the current study (section 3.2.3), the need to customize agile processes and practices to specific organizational contexts and needs was highlighted. This was suggested because there is seldom a one fit solution to all needs, and it is important that enterprises understand their own contextual constraints and requirements when opting to implement an agile culture. To further enhance such endeavours, it was also suggested that organisations leverage agile related technology toolsets to support the execution and management of their software engineering projects (section 3.3). For the current thesis, after evaluation of GZCS's realities and specific needs, the researcher opted to develop an IS prototype during the experimental component of this research.

The tool built was named GZ AgilePM Consolidator and ambitioned to support organizations in delivering quality agile projects and solutions to markets. This IS aimed to address the issue of conflicting information views generated by the large array of technology tools used by engineering teams such as the ones encountered at GZCS. The prototype was built as a web platform which aggregates data from disparate sources. It then presents this information through a unified portal with a consolidated view of agile related project information as a single source of truth. A key motivation behind its design as described earlier (figure 5.3), was that the combination of agile practices and processes, informed by a tool which would provide a consolidated information perspective, could enhance the management of agile projects, and hence have a ripple effect on the quality delivered to customers.

7.3.2 Describing the new activity system

In suggesting a move towards a practical agile oriented framework supported by appropriate technology, research objectives for this thesis and problems encountered at GZCS were aligned. In effect, case study subjects had identified management of the overwhelming amount of information they had to deal with as problematic. Most staff the researcher interacted with, believed that the current situation at the firm was plagued by siloed work which was mainly informed by the tools used individually according to project team functions. This in their view generated risk when feedback was trickled to project leadership and generated uncertainty at times from non-technical and technical staff alike.

For the researcher, this situation could also be contemplated through both a chain theory lenses as well as an Activity Theory perspective. In practice, some studies have suggested that software engineering in itself is analogous to a supply chain encompassing various activities centred around software development (Reddy 2009, p. 659). Other studies have even coined the term Software Focused Supply Chains (SFSCs) to highlight the importance of software related activities in the delivery of valuable products and services to customers (Tan & Yuan 2005, p. 187). For the purposes of the current research, delivering quality agile software projects and solutions could be described as the main value chain which is fed by central activities ranging from agile project management, to software and DevOps engineering. It is in the researcher's opinion hence vital that this chain be informed by robust technology which generates accurate information in a transparent fashion for project leadership and teams. Activity Theory provides a theoretical lens to further expand on GZCS' activity system after introduction of the current study's IS prototype. Chapter Four illustrated how AT was leveraged to illustrate the activity system studied in the course of this thesis (section 4.5.3). After implementation of the IS prototype, this activity system is as below:

Table 7-1 GZCS Activity System

Community	**Mediators**	**Activities**	**Tools**	**Outcomes**
It is still comprised of project stakeholders including project managers and engineers employed at GZCS in the context of agile software projects.	This still includes the competitive start-up environment scene GZCS operates in as well as an organizational culture focused around innovation and timely adoption of disruptive	These include development, DevOps and agile project management related activities and practices.	GZAPMC is the tool implemented to enhance agile projects delivery. It leverages data generated by the technology suites used at the firm to provide an accurate consolidated view of agile project information and hence helps better	Enhanced delivery of quality agile software projects and solutions.

	technology tools and agile practices.		inform project stakeholders and leadership.	

While essential components of the original activity system remain, the change introduced for purposes of this study revolved around one main IS as a central tool of the activity system. It is placed as a cornerstone of the new activity system to interface with all agile tooling used at the firm. Informed by a culture where agile processes are tailored to meet the organization's (section 5.3.1), the current study's prototype aimed to help the researcher meet enunciated objectives in Chapter One. Consequently, feedback from research subjects helped validate where these objectives were achieved or not.

According the study subjects, the implemented prototype has enhanced teams' ability to deliver quality agile projects and solutions because it informs stakeholders in an accurate manner and transparent fashion. For agile project managers and senior project leadership, introducing a web portal which provides a consolidated accurate information view from disparate data sources, has effectively reduced risks associated with work executed in siloes. In delivering a 360 view of all core software development, DevOps engineering and agile project management activities, the new portal helps engineers and project managers alike better estimate their time and understand the impact of activities conducted by project stakeholders. This according to GZCS leadership, helped improve the organization's delivery of quality agile projects and solutions; and for example, project managers and technical leads have been recently able to reduce gaps in timelines expectations and estimations thanks to the implemented prototype. This has helped GZCS teams reduce delays due to incomplete information and focus on delivering quality software projects quicker.

At this stage, the current chapter can now conclude by drawing from theory to propose an adjusted conceptual framework.

7.3.3 Emerging theoretical prospects

This study suggests a framework where an information portal is leveraged to consolidate agile related data from tools in use at a firm. The framework itself is largely informed by an AT based model. AT models have for example proven to be very useful in computer science and engineering research to inform IS design (Mwanza-Simwami 2001, sec. 4).

To design the proposed conceptual framework for this thesis, key elements of the organizational activity system were identified by leveraging Mwanza-Simwami's *8-Step Model* as described in table 5-1. Subsequently, the current research's AT inspired conceptual model, placed engineering (this includes both DevOps as well as software development) and agile project management activities at the centre of its basis.

Furthermore, conversations with senior management as well as project stakeholders at the firm led the researcher to understand how despite the adoption of an agile mindset as well as related enabling technologies, siloed usage of technologies tooling generated an overwhelming amount of project data in a scattered manner. Leveraging tools already in use, the researcher argued that this situation could be improved if stakeholders were provided with a consolidated information view as depicted in figure 5.3.

At this stage, following identification of the activities to be observed, an IS which leverages data from various toolset in use and aggregates this information into a meaningful portal for agile project stakeholders was deployed. In this new activity system, division of labour is inferred by the attribution of engineering and agile project management tasks amongst members of the community.

Data gathered and discussed in sections 6.2, 6.3 and 6.4 provided insight which supported the researcher's proposed framework. This seemed to be helped by the fact that most of this study's participants where overwhelming already comfortable with agile practices and supporting technologies. Except in fringe cases where some participants argued that the new proposed IS did not actually provide new raw project data points, most subjects saw tangible benefits in the aggregation of agile related data artefacts into one consolidated information portal. This was argued to provide one consistent source of truth for all stakeholders. Moreover, the tool was also perceived as beneficial to the organization in that it helped both engineering staff and project leadership in better informed decision making by reducing the risks of conflicting information and project data updates.

Consequently, to attempt repeatability of this thesis' experimental component in other organizational contexts, a refined conceptual framework was derived as illustrated in figure 7.2:

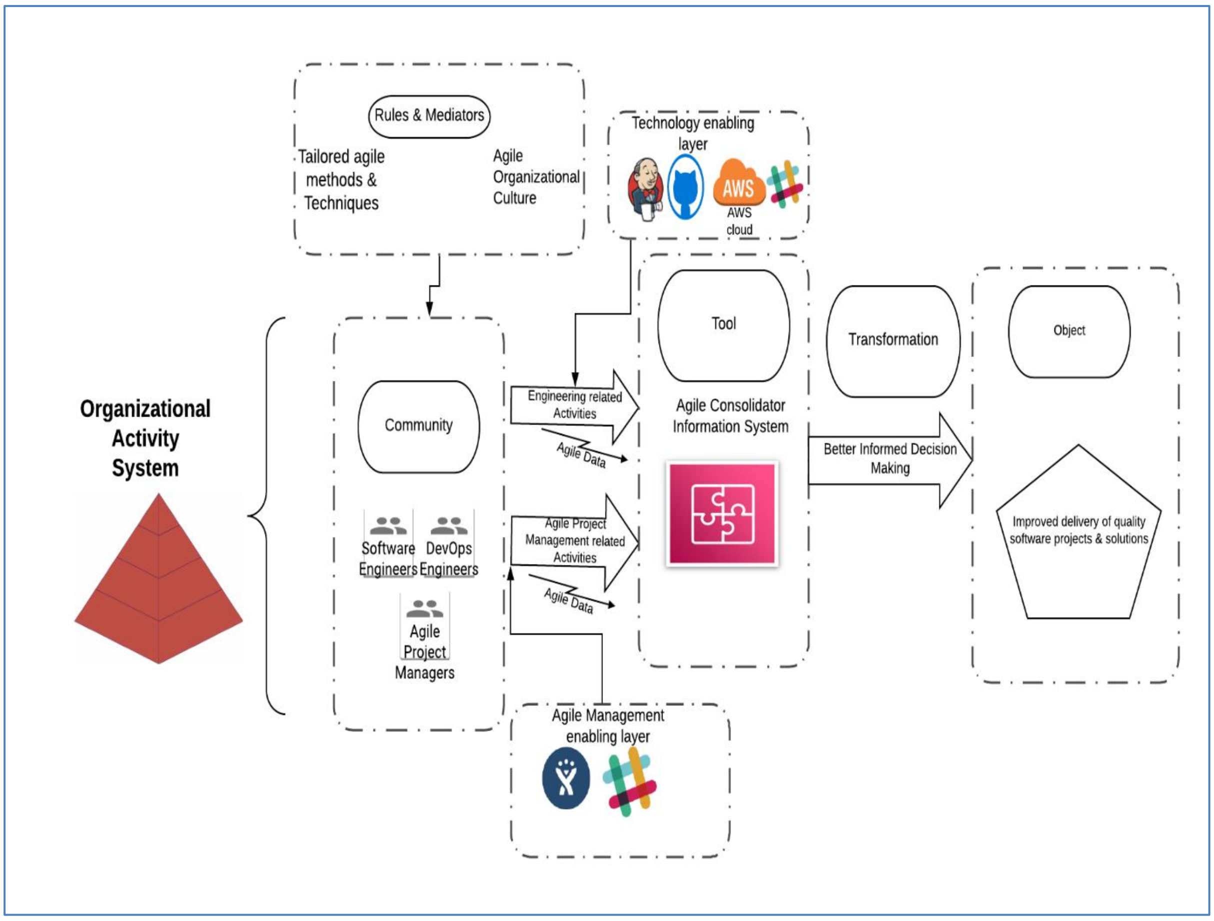

Rules & Mediators
Tailored agile methods & Techniques
Agile Organizational Culture
Technology enabling layer
AWS
AWS cloud
Organizational Activity System
Community
Software Engineers
DevOps Engineers
Agile Project Managers
Engineering related Activities
Agile Data
Agile Project Management related Activities
Agile Data
Tool
Agile Consolidator Information System
Transformation
Better Informed Decision Making
Object
Improved delivery of quality software projects & solutions
Agile Management enabling layer

Figure 7.2 Proposed framework for quality software projects delivery

The model aims to contribute to the body of knowledge by highlighting how important it is that organizations leverage technology tooling as well as related practices and techniques to facilitate the community's work in context of agile projects. This, coupled with a well-defined and shared agile organizational culture, helps inform how the community of members can collaborate in an effective manner to deliver agile projects for stakeholder and end users. Once this is effective, engineering and agile project management activities supported by enabling software suites can provide a large amount of helpful agile project data. Such data can at this stage be better managed and centralized into a central consolidated system which will help stakeholders gain an accurate meaningful source of truth. During the current research, this was found to enable better informed decision making and thus help improve the delivery of quality software projects and solutions.

7.4 Contribution of the study

As the thesis draws closer to being concluded, its modest contribution into information systems research as well as practice is mentioned before Chapter seven reaches its term.

7.4.1 Knowledge contribution

The current study featured an extensive exploration of academic sources in its literature survey. It nonetheless attempted to contextualize this literature review to practical cases by also drawing from industry reports and white papers. Furthermore, the experimental component of this research took place at a South African start-up. This was done to gather practical insights from firms involved with agile tooling as well as approaches for delivery of their software projects and solutions within sub-Saharan contexts.

Ultimately, the thesis contributed an expandable body of literature which leveraged AT as a theoretical lens. It consequently produced peer-reviewed academic articles on the management of agile software projects activities in practice. Researchers interested in finding means to use Activity Theory as an underlying framework to analyse modern IS phenomena could also benefit from the insight presented in this thesis as well as its proposed conceptual model for the delivery of quality agile projects.

7.4.2 Practical applications

An additional key contribution of this study was the creation of an IS prototype: GZAPMC. This tool could prove very useful to any organization involved with agile oriented commercial or open source tooling in their daily operations. In effect, the IS built for the purposes of this study can be plugged in to any existing infrastructure (agile software project tooling) and help project teams with a unified source of truth to better inform decision making and hence deliver quality agile software projects. It was designed and implemented with open source technologies and is furthermore customizable to specific needs should additional integration requirements arise. To achieve repeatability of similar solutions, it is suggested before hand that organizations identify their current activity system. Once this is achieved, the actors and processes/activities which could benefit from a framework like the one proposed in this study could be better understood. From a technical implementation standpoint, stakeholders need to identify the key systems which provide data relevant to the problem at hand and which require some integration. After this, all that is required is to find the relevant APIs and configure GZAPMC or a similar tool to meet their specific needs. A key assumption to keep in mind is that solutions such as the one proposed in this study require that tools in use provide some sort of interface to organizational project data. When this is not the case, it is advisable that leadership migrate towards open source solutions or proprietary cloud services which generally provide some sort of REST APIs into their backend systems.

7.5 Chapter Seven Summary

Chapter Seven re-introduced the context within which this research's case study was conducted. This was done to expand on the interpretation of research findings by exploring how GZCS staff perceived quality delivery and agility prior to implementation of the prototype.

The chapter then discussed validation of the AT model's outcome component (De Souza & Redmiles 2003, para. 6). After this, a discussion on the new activity system in place at the firm after implementation of said prototype followed. Exploration of GZAPMC' usage was then conducted to understand the organization's new activity system.

The chapter closed by describing a final draft of the proposed AT inspired conceptual framework and briefly discussed aspects of this study's practical contributions. The final chapter of this thesis concludes the research.

8 CHAPTER EIGHT: CONCLUSION AND RECOMMENDATIONS

8.1 Study overview

This study's key focus revolved around addressing issues modern organisations faced in the delivery of quality software projects and solutions to markets within agile contexts.

Chapter one introduced a background on challenges which software organisations faced starting with the software crisis of the 1960s. These were traditionally observed to be partly caused by the intangible elements of software products, the relative ease at which code can be duplicated, the ease at which amateur programmers can deliver functioning software and the fact that often, software integrity can be compromised as product design evolves. Although a disciplined approach to software engineering eased these concerns, modern firms are faced with persisting difficulties in delivering quality software projects and solutions to markets. In effect, present-day engineering teams are increasingly relying on agile methodologies, techniques and tools to help them consistently deliver quality software projects and solutions. This introductory portion of the dissertation concluded by enunciating the research problem at hand and derived research questions.

Chapters Two and Three proceeded with a literature review on topics such as software engineering, development processes, as well as software quality, its business value and related software quality models. Agile approaches, techniques, tools, methodologies and technologies were also discussed during the literature survey.

Chapter Four presented the research methodology and epistemological considerations which informed this study. The theoretical foundations, qualitative approach and interpretative paradigm leveraged for this research were also discussed.

Chapter Five described how an industry use case was conducted a South African software engineering start-up: GZ Consulting Services. A conceptual framework informed by Activity Theory was illustrated to underpin the practical component of the current research. This case study was based on the design and implementation of an information system prototype which aimed at consolidating agile and DevOps related information flows generated from the tools in use at GZ Consulting Services in the context of agile software engineering projects.

Finally, Chapters Six and Seven were written with the aim of exposing research findings and subsequent data analysis as well as interpretative lenses to readers.

These components of the thesis attempt to present a critical reading of data gathered and proceed to its interpretation. In the latter part of this research, the thesis aims to conclude with the study's limitations, possible future areas of research to explore as well as recommendations for local organisations operating in disruptive agile settings. Before this occurs, it would perhaps help to re-examine the study's stated objectives and discuss recommendations which might address the issues they raise.

8.2 Research questions revisited

The current study was underpinned by a central research question which could be re-formulated as follows:

"How can a modern organization facilitate delivery of quality agile software projects and solutions to markets through the leveraging of an agile framework supported by technology tooling?"

Considering that a case study would be conducted at a local South African engineering organization, and that such a framework would necessarily have to be informed by agile methodologies and related technologies, further sub-questions were derived in the form below:

- What does quality entail in software engineering projects and how does the modern software engineering industry deal with this issue?
- How can an organisation tailor its development processes in an agile manner and facilitate its delivery to market?
- What are the modern disruptive technological tools at the disposal of a South African small and medium enterprise (SME) to achieve the continuous delivery of quality software to market?
- How can an agile oriented information system complement these tools to facilitate an African SME's delivery of quality software to market?

To the first sub-question, this study highlighted how quality was difficult to define when discussing software. Nonetheless, the researcher understood software quality itself mainly as the ensemble of quality traits encompassed by the functional, structural, and process dimensions of software quality. The process dimension of quality software ties in with one of the current thesis's central assumptions which posits that improving the process by which software is produced will ultimately improve the software product's quality (section 6.2.3).

Therefore, modern organisations must understand the importance of agile methodologies and practices despite some important caveats.

8.3 Pre-Requisites to optimal agile implementation

Software engineering enterprises and teams need to understand that applying agile properly is heavily reliant on a shared understanding and mind frame by all stakeholders. Industry practitioners have described this interdependency between the required mind set for agile and the actual supporting processes and tools in what has been deemed an *Agile Onion* as described below:

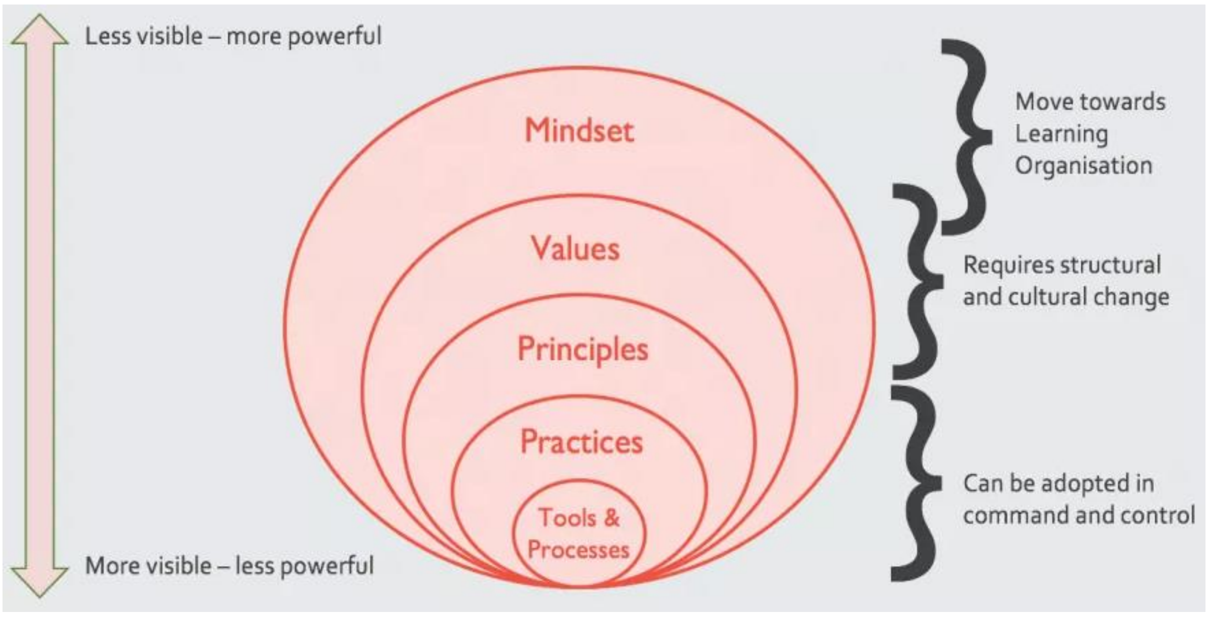

Less visible – more powerful
Mindset
Values
Principles
Practices
Tools & Processes
More visible – less powerful
Move towards Learning Organisation
Requires structural and cultural change
Can be adopted in command and control

Figure 8.1 Agile Onion (Jones 2019, p. 1)

What the diagram above highlights is that while agile tools and processes can be implemented with very little difficulties other than in terms of human resources as well as technical constraints, principles, values and the agile mindset tend to be more abstract and are less visible. These layers are nonetheless the most influential, and although some of them require deep cultural and structural evolutions, the agile mindset ultimately helps businesses transform into learning organizations (Doyle 2018, para. 6). It is hence important that this critical mindset be shared throughout an organization. Only then, can agile principles and practices with their supporting tools efficiently facilitate the delivery of quality software projects and solutions. Even then, firms rarely find a one fit solution to any random organizational settings. In such instances, some customization on specific agile practices and tooling might be required because in modern day settings, teams need to be increasingly adaptable. To this point, the second sub-question dealt with what happens when agile methodologies cannot randomly be implemented *stricto sensu* in all settings.

Once agile tools and processes have been adopted, engineering teams may yet experience difficulties in running their operations smoothly if their approach is not customized to the environment, they operate in. When this occurs, it is important for agile teams to tailor processes and tools for their specific needs. In such instances, as previously mentioned (table 3-7), project teams need to focus on improving team coordination, scale projects as required and encourage the adoption of novel practices. Some industry insight has for instance suggested that organization focus on blending agile techniques for their specific needs, establish a clear agile communication structure, leverage traditional SDLC processes on an as needed basis, and still put an emphasis on flexibility (Reichert 2016, p. 1). At GZCS, this was essentially done by adopting a mix of both Kanban and some Scrum practices. Nonetheless, at other times GZCS engineering teams leveraged XP practices such as Pair Programming when onboarding new software developers for instance.

Ultimately, these practices are even further enhanced when supported by enabling technology including novel cloud services, as well as related agile tooling. For this purposes, the next section revisits subsequent research questions.

8.4 Revisiting the case study

The third sub-question discussed technological platforms while the fourth one dealt with how to complement some of these existing tools in practical industry setting. In the context of start-up environments such as the one explored during the current case study, cloud computing solutions were identified as a game changer for engineering teams with no or little capital. To effectively run their software projects, firms such as GZCS leveraged agile approaches and processes enhanced by cloud technology services in terms of source control management, infrastructure deployment, internal team communication and overall project management. These positive aspects were noted with some risk nonetheless when technology tooling was used in uncoordinated siloes as explored through the fourth sub-question.

From a software project lifecycle perspective, leveraging innovative agile software services has been a huge enhancement for organizations. Software suites such as Atlassian’s JIRA, GITHUB.com, Jenkins and more have facilitated the adoption and practices of agile practices for teams. Nonetheless, often, the siloed usage of these tools has yielded a large amount of disparate information sources which can lead to inaccurate and incomplete assumptions from different stakeholders. This risk could exponentially increase as the array of agile practices and tooling augments. Consequently, for the current study, a prototyped IS, GZAPMC was built to aggregate information generated by agile technology tooling in a consolidated fashion via a web portal. This was considered beneficial by project stakeholders as it presented once source of truth for project leadership and engineering staff to better inform their decision making and enhance their ability to deliver quality software projects and solutions to markets. Consequently, facilitating this delivery of quality software projects informed the current's research Activity Theory oriented proposed framework. In the next sections, limitations to this model are given a glimpse.

8.5 Limitations

A key limitation of the current thesis was in the experimental component of the research. In effect, single case studies are often suspected in terms of a perceived lack of methodological rigour (Willis 2014, para. 15). Subjectivity is also often questioned in research which leverages case studies and qualitative approaches in general (Berger & Lune in Willis, 2014 para. 16). To mitigate this, the researcher used a practical industry case study. This partly consisted in the implementation of an IS prototype for study subjects to evaluate. Gathering empirical evidence based around usage of this prototype was deemed to give the research scientific grounding. Furthermore, leveraging an interpretative lens coupled with an AT informed theoretical framework was the underpinning academic thread for this thesis.

Another limitation of the research resides in the final conceptual framework itself. Considering that the current research drew some of its assumptions from Chain theory, very little specifics on customization of agile processes emerged during the experimental portion of the industrial case study. The essential elements of this customization processes were hence mostly dealt with in the literature review informed by industry case studies and research.

Finally, users had many suggestions to improve the experimental IS. These ranged from UX and front-end enhancements, to additional communication capabilities as well as a broader range of integrations or plug-ins. These were unfortunately not all addressed due to time constraints, but the researcher mitigated these shortfalls by ensuring the capabilities at hand helped deal with enunciated research objectives. Most of the limitations encountered were mitigated, although a degree of scientific scepticism or rather, critical thinking, should always be observed in exploring research findings. Ultimately, to conclude this thesis, the following sections attempt to point at tentative directions which might yield interesting research interests.

8.6 Further areas to look at

Findings explored in the current study have highlighted the importance of paying to challenges in delivering quality agile software projects and solutions. To this end, a couple of other subjects would perhaps be worthy of academic research.

As mentioned previously (section 3.2.3), organizations implementing agile practices need to focus on customizing processes to their specific needs. It would perhaps be fitting to further explore how this selection methodology of modern practices is practically implemented in industry settings.

Additionally, there is a need to eventually address improvements on GZAPMC suggested by users. On top of these improvements, it would also be perhaps be interesting to add statistical measurements capabilities to the tool. If these capabilities were to be informed by domains such as CMMI or SQA, measurements of tangible code quality benefits or improvements could be achieved.

Finally, in the quest to deliver quality software projects and artefacts, it would also be fitting to explore more novel areas of research such as DataOps. This is argued to integrate agile engineering practices as well as statistical process controls to deliver relevant analytics and business intelligence to organizations (DataKitchen 2018, fig. 2). Taking into account how the current research's AT inspired conceptual framework placed software, DevOps engineering as well as agile project management as its central observable activities; DataOps exploration offers researchers an opportunity to determine how Lean Manufacturing can also be added to the mix to provide actionable business intelligence to maintain a competitive edge in the current knowledge based economy.

REFERENCES

Acton, D., Kourie, D. G. and Watson, B. W. (2014) ‘Quality in software development : A pragmatic approach using metrics’, *South African Computer Journal*, (52), pp. 1–12.

Adu, P. (2016) *Step-by-step Process of Conducting Qualitative Analysis Using NVivo 11*, *SlideShare*. Available at: https://www.slideshare.net/kontorphilip/stepbystep-process-of-conducting-qualitative-analysis-using-nvivo-11 (Accessed: 15 June 2018).

Adu, P. (2018) *What to do with your Data: Qualitative Research (Dr. Philip Adu) - YouTube*. Youtube. Available at: https://www.youtube.com/watch?v=QnbX55u1J7k (Accessed: 5 November 2018).

Ahmad, N. *et al.* (2013) ‘Requirements analysis of android application using activity theory: A case study’, *2013 International Conference of Information and Communication Technology, ICoICT 2013*. IEEE, pp. 145–149. doi: 10.1109/ICoICT.2013.6574563.

Akbar, R. *et al.* (2014) ‘Software development process tailoring for small and medium sized companies’, *2014 International Conference on Computer and Information Sciences (ICCOINS)*, pp. 1–5. doi: 10.1109/ICCOINS.2014.6868453.

Alajrami, S., Gallina, B. and Romanovsky, A. (2016) ‘EXE-SPEM : Towards Cloud-based Executable Software Process Models’, in *4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. 4th edn. Rome, pp. 1–10. Available at: http://0-ieeexplore.ieee.org.oasis.unisa.ac.za/stamp/stamp.jsp?tp=&arnumber=7954401.

Ambler, S. and Lines, M. (2012) *Disciplined Agile Delivery: A Practionner’s Guide to Agile Software Delivery in the Enterprise*. 1st edn. Edited by Pearson. Boston: IBM Press.

Anderson, D. (2003) *Agile management for software engineering : applying the theory of constraints for business results*, *The Coad series*. Prentice Hall PTR. Available at: http://www.loc.gov/catdir/toc/ecip047/2003017798.html.

Antony, J. (2004) ‘Some pros and cons of six sigma: An academic perspective’, *TQM Magazine*, 16(4), pp. 303–306. doi: 10.1108/09544780410541945.

Armenise, V. (2015) ‘Continuous Delivery with Jenkins: Jenkins Solutions to Implement Continuous Delivery’, *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, pp. 24–27. doi: 10.1109/RELENG.2015.19.

Ashrafi, N. (2003) ‘The impact of software process improvement on quality: In theory and practice’, *Information and Management*, 40(7), pp. 677–690. doi: 10.1016/S0378-7206(02)00096-4.

Bakal, M. (2012) *DevOps : Extending Agile Development Disciplines to Deployment*. Available at: https://www-01.ibm.com/events/wwe/grp/grp004.nsf/vLookupPDFs/Extending_Continuous_Delivery_Disciplines/$file/Extending_Continuous_Delivery_Disciplines.pdf.

Balalaie, A., Heydarnoori, A. and Jamshidi, P. (2016) 'Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture', *IEEE Software*, 33(3), pp. 42–52. doi: 10.1109/MS.2016.64.

Baller, S., Dutta, S. and Lanvin, B. (2016) *The Global Information Technology Report 2016 The Global Information Technology Report 2016 Innovating in the Digital Economy*.

Bang, S. K. *et al.* (2013) 'A grounded theory analysis of modern web applications for DevOps', *Proceedings of the 2nd annual conference on Research in information technology - RIIT '13*, (October 2013), p. 61. doi: 10.1145/2512209.2512229.

Basili, V. and Caldiera, G. (1995) 'Improve Software Quality by Reusing Knowledge and Experience.pdf', *SLOAN Management Review*, (Spring). Available at: http://www.cs.umd.edu/~basili/publications/journals/J55.pdf.

Baumer, E. P. S. and Tomlinson, B. (2011) 'Comparing Activity Theory with Distributed Cognition for Video Analysis: Beyond "Kicking the Tires"', in *29th CHI Conference on Human Factors in Computing Systems*. Vancouver, pp. 133–142. doi: 10.1145/1978942.1978962.

Beckwitt, A. (2016) *Qualitative Data Analysis Coding Qualitative Data Qualitative Dissertation | Asher Consult*, *http://asherconsult.com*. Available at: http://asherconsult.com/what-are-inductive-and-deductive-approaches-to-coding-qualitative-data/ (Accessed: 18 October 2018).

Beizer, B. (1997) 'Cleanroom process model: a critical examination', *IEEE Software*, 14(2), pp. 14–16. doi: 10.1109/52.582968.

Bell, T. E. and Thayer, T. A. (1976) 'Software Requirements : Are They Really a Problem ?', *International Conference on Software Engineering*, pp. 61–68.

BER (2016) *The small, medium and micro enterprise sector of South Africa*, *Bureau for Economic Research*. Stellenbosch. Available at: http://www.seda.org.za/Publications/Publications/The Small, Medium and Micro Enterprise Sector of South Africa Commissioned by Seda.pdf.

Bernstein, L. and Yuhas, C. . (2005) *Trustworthy Systems Through Quantitative Software Engineering*. 1st edn. Hoboken: Wiley.

Bertham, A. (2018) *Is 'Continuous Delivery with Docker and Jenkins' right for you?*, *TechTarget Search ITOperations*. Available at: https://searchitoperations.techtarget.com/feature/Is-Continuous-Delivery-with-Docker-and-Jenkins-right-for-you?src=5821488&asrc=EM_ERU_103655046&utm_content=eru-rd2-rcpE&utm_medium=EM&utm_source=ERU&utm_campaign=20181119_ERU Transmission for 11/19/2 (Accessed: 22 November 2018).

Bessin, G. (2004) *The Business value of software quality*. Available at: http://www.ibm.com/developerworks/rational/library/dec04/bessin/ (Accessed: 23 August 2013).

Bhasin, S. (2012) 'Quality assurance in agile: A study towards achieving excellence', *Proceedings - Agile India 2012, AgileIndia 2012*, pp. 64–67. doi: 10.1109/AgileIndia.2012.18.

Boer, A. and Sileno, G. (2013) *AGILE : a methodology for Advanced Governance of Information services through Legal Engineering*. Amsterdam: Uva-DARE (Digital Academic Repository). Available at: https://pure.uva.nl/ws/files/1735434/156091_agile_final.2013.pdf (Accessed: 23 March 2018).

Bot, H. (2019) *Through the MicroScope: How to escape the microservice dilemma with WSO2 API Microgateway*, *Yenlo Knowledge Blog*. Available at: https://www.yenlo.com/blog/through-the-microscope-how-to-escape-the-microservice-dilemma-with-wso2-api-microgateway?utm_source=hs_email&utm_medium=email&utm_content=71257464&_hsenc=p2ANqtz--ZUucRE19oacZCEAN2mnfOv2CqqS9XG2L21zRGY26E8uYtMHKYM3SIUgZMWofdIbrk (Accessed: 30 March 2019).

Brady, R. M., Anderson, R. J. and Ball, R. C. (1999) *Murphy's law, the fitness of evolving species, and the limits of software reliability*. Cambridge. doi: UCAM-CL-TR-471.

Brennecke, A. *et al.* (1996) 'History of Software Engineering'.

Brinkkemper, S. (1996) 'Method engineering: Engineering of information systems development methods and tools', *Information and Software Technology*, 38(4 SPEC. ISS.), pp. 275–280. doi: 10.1016/0950-5849(95)01059-9.

Bryant, S., Boulay, B. and Romero, P. (2006) *XP and pair programming practices*, *PPIG News Letter*. Available at: https://www.researchgate.net/publication/231167972_XP_and_pair_programming_practices

(Accessed: 28 July 2019).

Buchanan, I. (2018) *Agile and DevOps - Friends or Foes? | Atlassian*, *www.atlassian.com*. Available at: https://www.atlassian.com/agile/devops (Accessed: 5 June 2018).

Caroll, J., Bertelsen, O. and Bødker, S. (2003) *Activity Theory, HCI Models, Theories and Frameworks: Toward an Interdisciplinary Science*. Edited by M. Kaufmann. Elsevier Ltd. Available at: http://ci.nii.ac.jp/naid/10007611272/.

Casare, S. *et al.* (2016) 'Meduse: an Approach for Tailoring Software Development Process', in *2016 21st International Conference on Engineering of Complex Computer Systems*. Kuala Lampur, p. 197. doi: 10.1109/ICECCS.2016.033.

Cervera, M. *et al.* (2011) 'Turning method engineering support into reality', *IFIP Advances in Information and Communication Technology*, 351 AICT, pp. 138–152. doi: 10.1007/978-3-642-19997-4_14.

Chacon, S. and Straub, B. (2014) *Pro Git*. 2nd Editio. Edited by Apress. Mountain View: Creative Commons. Available at: https://git-scm.com/book/en/v2.

Chappell, D. (2012a) *The Business Value of Software Quality*. Available at: http://www.davidchappell.com/writing/white_papers/The_Business_Value_of_Software_Quality-v1.0-Chappell.pdf.

Chappell, D. (2012b) *THE THREE ASPECTS OF SOFTWARE QUALITY : FUNCTIONAL , STRUCTURAL , AND PROCESS Sponsored by Microsoft Corporation*. Available at: http://davidchappell.com/writing/white_papers/The_Three_Aspects_of_Software_Quality_v1.0-Chappell.pdf.

Chassin, M. R. (1998) 'Is Health Care Ready for Six Sigma Quality?', *Milbank Quarterly*, 76(4), pp. 565–591. doi: 10.1111/1468-0009.00106.

Christian, R. (2016) *Portfolio for JIRA meets kanban*, *https://www.atlassian.com*. Available at: https://www.atlassian.com/blog/jira-software/kanban-with-portfolio-for-jira (Accessed: 5 February 2017).

Christof, E. *et al.* (2016) 'DevOps', *IEEE Software*, 33(May-June), pp. 94–100.

Cohen, D. J. (2008) *Assessing the business value of software process improvement using CMMI ® in South Africa*. University of Pretoria.

Cois, C. A. (2014) *DevOps and Agile Development A VMware Field Perspective*. Available at: https://insights.sei.cmu.edu/sei_blog/2014/11/devops-and-agile.html.

Collins-Sussman, B., Fitzpatrick, B. W. and Pilato, C. M. (2008) *Version Control with Subversion (Compiled from r3305)*, *ReVision*. doi: 10.1081/E-ELIS3-120044663.

Cooper, R. G. (2016) 'Agile-stage-gate hybrids', *Research Technology Management*, 59(1), pp. 21–29. doi: 10.1080/08956308.2016.1117317.

Cossentino, M. and Seidita, V. (2005) 'Composition of a New Process to Meet Agile Needs Using Method Engineering', *Software Engineering for Multi-Agent Systems III*, 3390, pp. 36–51. doi: 10.1007/978-3-540-31846-0_3.

Creswell, J. (1998) *Qualitative Inquiry and Research Design: Choosing Among Five Traditions*. London: SAGE publications Inc.

Cwele, C. (2016) 'National Integrated ICT Policy White Paper', *Government Gazette*, 1212(40325), pp. 4–176.

Dalling, T. (2009) *SOLID Class Design: The Liskov Substitution Principle*, *www.tomdalling.com*. Available at: https://www.tomdalling.com/blog/software-design/solid-class-design-the-open-closed-principle/ (Accessed: 22 April 2015).

Daly, L. (2018) *Kanplan: where your backlog meets kanban*, *https://www.atlassian.com*. Available at: https://www.atlassian.com/agile/kanban/kanplan (Accessed: 27 May 2018).

DataKitchen (2018) *DataOps is NOT Just DevOps for Data*, *Medium*. Available at: https://medium.com/data-ops/dataops-is-not-just-devops-for-data-6e03083157b7 (Accessed: 15 January 2019).

Dawson, B. (2017) *What Is DevOps?*, *DZone DevOps*. Available at: https://dzone.com/articles/what-is-devops-4 (Accessed: 23 September 2018).

Deek, F., McHugh, J. and Eljabiri, O. (2005) *Strategic Software Engineering: An Interdisciplinary Approach*. 1st edn. Auerbach Publications.

Deming, E. (1993) *The New Economics: For industry, government, and education*. 2nd edn. Cambridge: The MIT Press.

Dervin, F. *et al.* (2016) *Constructing Methodology for Qualitative Research Researching Education and Social Practices*. Palgrave Macmillan. doi: 10.1057/978-1-137-59943-8.

Desfray, P. and Raymond, G. (2014) *Modeling enterprise architecture with TOGAF : a practical guide using UML and BPMN*. 1st edn. Edited by M. Kaufmann. Waltham: Morgan Kaufmann. Available at: https://books.google.com.br/books?id=b6L0AwAAQBAJ&pg=PA259&lpg=PA259&dq=uso+do+UML+com+Togaf&source=bl&ots=8bxk0U2DGR&sig=ACfU3U21m80VgzMuqaFyiTzJywNwPOHsDQ&hl=pt-BR&sa=X&ved=2ahUKEwjNvYzw_r7hAhXMErkGHXnMAWkQ6AEwDHoECAgQAQ#v=onepage&q=uso do UML com Togaf.

DiCicco-Bloom, B. and Crabtree, B. (2006) 'The qualitative research interview', *Medical Education*, 40(4), pp. 314–321. doi: 10.1111/j.1365-2929.2006.02418.x.

Doyle, C. (2018) *THE AGILE ONION – LAYER BY LAYER*, *https://assurity.nz*. Available at: https://assurity.nz/insights/the-agile-onion-layer-by-layer/ (Accessed: 10 December 2018).

DTPS (2018) *Establishment of the presidential commission on the fourth industrial revolution*, *Government Gazette*. doi: http://dx.doi.org/9771682584003-32963.

Dwolatzky, B. and Trengove, E. (2002) *A Software Development Process Applicable to 'Information Age' Applications*. doi: RES/SC/02/18621.

Dybå, T. *et al.* (2011) 'Qualitative research in software engineering', *Empirical Software Engineering*, 16(4), pp. 425–429. doi: 10.1007/s10664-011-9163-y.

Dyck, A., Penners, R. and Lichter, H. (2015) 'Towards definitions for release engineering and DevOps', *Proceedings - 3rd International Workshop on Release Engineering, RELENG 2015*, p. 3. doi: 10.1109/RELENG.2015.10.

EMA (2018) *IBM Delivers Innovative Cloud Infrastructure Solutions Spanning On-Premises & Hybrid Cloud*. Available at: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=POW03195USEN (Accessed: 1 March 2018).

Fereday, J. and Muir-Cochrane, E. (2006) 'Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development', *International Journal of Qualitative Methods*, 5(1), pp. 80–92. doi: 10.1177/160940690600500107.

Fitzgerald, B. (2012) 'Software crisis 2.0', *Computer*, 45(4), pp. 89–91. doi: 10.1109/MC.2012.147.

Fitzgerald, B., Hartnett, G. and Conboy, K. (2006) 'Customising agile methods to software practices at Intel Shannon', *European Journal of Information Systems*, 15(2), pp. 200–213. doi: 10.1057/palgrave.ejis.3000605.

Fitzgerald, B. and Stol, K. J. (2017) ‘Continuous software engineering: A roadmap and agenda’, *Journal of Systems and Software*. Elsevier Ltd., 123(July), pp. 176–189. doi: 10.1016/j.jss.2015.06.063.

FoodRisC (2016) *Mixed methods research*, *http://resourcecentre.foodrisc.org*. Available at: http://resourcecentre.foodrisc.org/mixed-methods-research_185.html (Accessed: 29 July 2019).

Fowler, M. (2006) ‘Continuous Integration’, *Integration The Vlsi Journal*, 26(1), pp. 1–14. doi: 10.1007/978-1-4302-0142-7_9.

Fox, S., Shields, T. and Pang, S. (2012) *Five Steps to Choosing an Enterprise-Class Cloud Service Provider*. WP-7154. Available at: http://cstor.com/wp-content/uploads/2015/03/wp-7154.pdf.

Gaaloul, K. and Molnar, W. (2014) ‘Research methodologies in enterprise engineering: Insights from a workshop’, in *Proceedings - 2014 International Workshop on Advanced Information Systems for Enterprises, IWAISE 2014*. Tunis, pp. 58–64. doi: 10.1109/IWAISE.2014.8.

Gaoussou, A. K. D. and Mnkandla, E. (2018) ‘Facilitating the Management of Agile and Devops Activities: Implementation of a Data Consolidator’, in *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*. IEEE, pp. 1–6. doi: 10.1109/ICABCD.2018.8465451.

García, F. *et al.* (2007) ‘Managing software process measurement: A metamodel-based approach’, *Information Sciences*, 177(12), pp. 2570–2586. doi: 10.1016/j.ins.2007.01.018.

Gartner (2010) *Gartner Identifies Top 30 Countries for Offshore Services in 2010-2011*. Available at: https://www.gartner.com/newsroom/id/1500514 (Accessed: 12 December 2016).

Gartner (2015) *Gartner says IT spending in South Africa to reach $26.6bn in 2016, up 5% from 2015*, *ITWeb Financial, Gartner Sponsored content*. Available at: http://www.itweb.co.za/index.php?option=com_content&view=article&id=146241 (Accessed: 12 December 2016).

Gartner (2018) *Gartner for Technical Professionals analysts answer pressing questions in their areas of expertise*, *Gartner*. Available at: https://www.gartner.com/en/conferences/emea/catalyst-uk/why-attend/event-resources/analyst-qa?utm_campaign=EVT_EMEA_2018_CATUK6_BB_E6_Prospect&utm_medium=email&utm_source=Eloqua&cm_mmc=Eloqua-_-Email-_-LM_EVT_EMEA_2018_CATUK6_BB_E6_Prospect-_-0000 (Accessed: 12 July 2018).

Gedera, D. S. P. and Williams, P. J. (2016) *Activity theory in education: Research and practice*, *Activity Theory in Education: Research and Practice*. Rotterdam/Boston/Taipei: Sense Publishers.

Ghanim, Y. (2016) ‘Toward a Specialized Quality Management Maturity Assessment Model’, *Proceedings of the 2nd Africa and Middle East Conference on Software Engineering - AMECSE ’16*, pp. 1–8. doi: 10.1145/2944165.2944166.

Gibbs, W. W. (1994) ‘Software’s Chronic Crisis’, *Scientific American*, 271(3), pp. 86–95. doi: 10.1038/scientificamerican0994-86.

Giblin, M. (2012) *Introducing agile methods in a large software development organisation: a case study*. University of Limerick. Available at: http://ulir.ul.ie/handle/10344/2477.

Glazer, H. *et al.* (2008) ‘CMMI ® or Agile : Why Not Embrace Both !’, *(CMU/SEI-2008-TN-003). Software Engineering Institut*, (November), p. 48. doi: 10.1109/AGILE.2006.30.

Gray, D. E. (2013) ‘Theoretical Perspectives and Research Methodologies’, in *Doing Research in the Real World*. 3rd editio. Greenwich: SAGE publications, p. 422. doi: 10.1017/CBO9781107415324.004.

Gumbi, L. N. and Mnkandla, E. (2015) ‘Investigating South African Vendors ’ Cloud Computing Value Proposition to Small , Medium and Micro Enterprises : A Case of the City of Tshwane Metropolitan Municipality’, *The African Journal of Information Systems*, 7(4), pp. 1–17.

Happiest-Minds (2014) *DevOps*. Available at: http://www.happiestminds.com/whitepapers/devops.pdf.

Hardman, J. (2005) ‘Activity theory as a potential framework for technology research in an unequal terrain’, *Sajhe*, 19(2), pp. 378–392. doi: 10.1017/CBO9781107415324.004.

Harmsen, F., Brinkkemper, S. and Oei, H. (1994) *Situational Method Engineering for Information System Project Approaches*, *Information Systems*.

Harrell, M. and Bradley, M. (2009) *Data Collection Methods. Semi-Structured Interviews and Focus Groups*. Santa Monica. doi: RAND.

Haryono, K. (2015) ‘The extreme programming approach for financial management system on local government’, *2015 International Conference on Science and Technology (TICST)*, pp. 29–34. doi: 10.1109/TICST.2015.7369335.

Hasan, H. and Kazlauskas, A. (2014) ‘Activity Theory: who is doing what, why and how’, *Being Practical with Theory: A Window into Business Research*, pp. 9–14. doi: 10.3102/0034654306298273.

Hashemnezhad, H. (2015) ‘Qualitative Content Analysis Research : A Review Article’, *Journal of ELT and Applied Linguistics (JELTAL)*, 3(1), pp. 54–62.

Hashim, N. H. and Jones, M. L. (2007) ‘Activity Theory : A framework for qualitative analysis’, *Qualitative Research*, pp. 3–5. Available at: http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1434&context=commpapers%5Cnhttp://ro.uow.edu.au/cgi/viewcontent.cgi?article=1434&amp;context=commpapers.

Head, G. E. (1994) ‘Six-Sigma Software Using Cleanroom Software Engineering Techniques’, *Hewlett-Packard Journal*, 1994(June), pp. 40–50.

Heale, R. and Forbes, D. (2013) ‘Understanding triangulation in research Understanding triangulation in research References Email alerting service’, 16(4), pp. 16–17. doi: 10.1136/eb-2013-101494.

Heimerl, F. *et al.* (2014) ‘Word cloud explorer: Text analytics based on word clouds’, *Proceedings of the Annual Hawaii International Conference on System Sciences*. IEEE, pp. 1833–1842. doi: 10.1109/HICSS.2014.231.

Helen, S. and Tracy, H. (2016) ‘Agile Processes, in Software Engineering, and Extreme Programming’, in *17th International Conference*. Edinburgh: Springer International Publishing, p. 334. doi: 10.1007/978-3-319-33515-5_13.

Henderson-Sellers, B. and Serour, M. (2005) ‘Creating a Dual-Agility Method : The Value of Method Engineering’, *Journal of Database Management*, 16(4).

Henderson, F. (2017) *Software Engineering at Google*. Available at: https://arxiv.org/ftp/arxiv/papers/1702/1702.01715.pdf.

Higgs, P. (2010) ‘Towards an indigenous African epistemology of community in education research’, *Procedia - Social and Behavioral Sciences*, 2(2), pp. 2414–2421. doi: 10.1016/j.sbspro.2010.03.347.

Hirschheim, R. A. (1992) ‘Information Systems Epistemology: An Historical Perspective’, *Information systems research: Issues, methods and practical guidelines*, pp. 9–33. doi: 10.1017/CBO9781107415324.004.

Hoffer, J. A., George, J. F. and Valacich, J. S. (2016) *Modern systems analysis and design*. Pearson.

Homann, U. and Mckenty, J. (2018) ‘Application Modernization with Cloud Flexibility’. Pivotal, pp. 1–16. Available at: https://content.pivotal.io/white-papers/application-modernization-with-cloud-flexibility.

Hunt, J. (2006) *Agile Software Construction (Google eBook)*. 1st editio. Springer. Available at: http://books.google.com/books?id=P8sWloUWrdIC&pgis=1.

Hurtado Alegría, J. A. *et al.* (2011) 'An MDE approach to software process tailoring', *Proceeding of the 2nd workshop on Software engineering for sensor network applications - SESENA '11*, (June 2014), p. 43. doi: 10.1145/1987875.1987885.

Hussain, A. and Mkpojiogu, E. O. C. (2015) 'Application of the Iso / Iec 25010 Standard in of an Online Health Awareness System', *Jurnal Teknologi*, 5, pp. 9–13. doi: 10.11113/jt.v77.6107.

Hutterman, M. (2012) *DevOps for Developers*. Edited by B. Renow-Clarke. New York: Apress.

Hyatt, L. and Rosenberg, L. (1996) 'A Sofware Quality Model And Metrics For Risk Assessment', in *Product Assurance Symposium and Software Product Assurance Workshop*. Noordwijk: European Space Agency, pp. 209–2012.

Iadis (2012) 'IADIS INTERNATIONAL CONFERENCE INFORMATION SYSTEMS 2012', in Nunes, M. B., Isaias, P., and Powell, P. (eds) *IADIS INTERNATIONAL CONFERENCE INFORMATION SYSTEMS*. Berlin: IADIS, International Association for development of the information society, p. 460.

Iain, H. (2013) 'Using questionnaires in qualitative research', in *Qualitative Research Methods in Human Geography*, pp. 191–216. Available at: http://ro.uow.edu.au/sspapers/2518 (Accessed: 6 December 2018).

IDC (2015) *ICT Spending in South Africa to Top $ 26 . 6 Billion in 2016 as Digital Transformation Initiatives Take Hold | About IDC | IDC CEMA ICT Spending in South Africa to Top $ 26 . 6 Billion in 2016 as Digital Transformation Initiatives Take Hold*. Available at: http://idc-cema.com/eng/about-idc/press-center/63187-ict-spending-in-south-africa-to-top-26-6-billion-in-2016-as-digital-transformation-initiatives-take-hold (Accessed: 12 December 2016).

ILDP (2014) *Informal, Small Medium and Micro Enterprise (SMME) Retailers in South Africa*. Available at: http://www.wrseta.org.za/ILDP_2014/Syndicate 1- Ratoon.pdf.

IODSA (2016) *King IV Report Draft 2016, 2016*.

ISO/IEC and IEEE (2010) 'ISO/IEC/IEEE 24765:2010 - Systems and software engineering -- Vocabulary', *Iso/Iec Ieee*, 2010, p. 410. doi: 10.1109/IEEESTD.2010.5733835.

ITU (2016) *A review of Micro, Small and Medium Enterprises in the ICT Sector 2016*. 2016th edn. ITU. Available at: https://www.itu.int/dms_pub/itu-s/opb/gen/S-GEN-EMERGE.01-2016-PDF-E.pdf.

Jain, P., Sharma, A. and Ahuja, L. (2018) ‘The Impact of Agile Software Development Process on the Quality of Software Product’, *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. IEEE, pp. 812–815. doi: 10.1109/ICRITO.2018.8748529.

Javadi, M. and Zarea, K. (2016) ‘Understanding Thematic Analysis and its Pitfall’, *Journal of Client Care*, 1(1), pp. 34–40. doi: 10.15412/J.JCC.02010107.

Jones, C. (2010) *SOFTWARE QUALITY IN 2010 : A SURVEY OF THE STATE OF THE ART*. Available at: http://www.sqgne.org/presentations/2010-11/Jones-Nov-2010.pdf.

Jones, R. (2019) *The Agile Radar: An Approach for Understanding Agile - DZone Agile*, *Agile Zone*. Available at: https://dzone.com/articles/agile-radar-an-approach-for-understanding-what-is?edition=473195&utm_source=Weekly Digest&utm_medium=email&utm_campaign=Weekly Digest 2019-04-10 (Accessed: 10 April 2019).

Joseph, N., Marnewick, C. and Jan Santana, M. (2016) ‘Agile Software Development and IT Project Performance in South Africa: A positive relationship?’, in. Orlando, Florida, USA: International Association for Management of Technology, pp. 338–358.

Joshua, C. E., Nehemiah, M. and Ernest, M. (2015) ‘A Conceptual Culture-Oriented e-Learning System Development Framework (e-LSDF): A Case of Higher Education Institutions in South Africa’, *International Journal of Trade, Economics and Finance*. doi: 10.18178/ijtef.2015.6.5.479.

Juopperi, M. (2017) *Deployment automation with ChatOps and Ansible*. Helsinki Metropolia University of Applied Sciences. Available at: https://www.theseus.fi/bitstream/handle/10024/127446/Deployment automation with ChatOps and Ansible.pdf?sequence=1.

Just, S. *et al.* (2016) ‘Switching to Git: The Good, the Bad, and the Ugly’, *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pp. 400–411. doi: 10.1109/ISSRE.2016.38.

Kalermo, J. and Rissanen, J. (2002) *Agile software development in theory and practice*, *University of Jyväskylä*. University of Jyväskylä. Available at: https://ebe47aaf-a-62cb3a1a-s-sites.googlegroups.com/site/manishsharma78/home/advanced-software-engineering/ASE.pdf?attachauth=ANoY7cpqt2hNwxlAfFtMJts74mhdJMQyI4bfNmkcbti95L3tyC97U-KOI0Fy6qnqIv1GBCmeTT5t1j4wZJbSmbnKAP6Tg-4W5cXHn7FdqAKOmuIMJ_FohV2vXev2wpuQ (Accessed: 2 January 2019).

Kalus, G. and Kuhrmann, M. (2013) ‘Criteria for software process tailoring: a systematic review’, *Proceedings of the 2013 International Conference on Software and System Process - ICSSP 2013*, p. 171. doi: 10.1145/2486046.2486078.

Kamuto, M. B. and Langerman, J. J. (2018) ‘Factors inhibiting the adoption of DevOps in large organisations: South African context’, *RTEICT 2017 - 2nd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, Proceedings*, 2018-Janua, pp. 48–51. doi: 10.1109/RTEICT.2017.8256556.

Kan, S. (2002) *Metrics and Models in Software Quality Engineering*. 2nd edn. Addison Wesley.

Kaptelinin, V. and Nardi, B. (2007) ‘Acting with technology: Activity theory and interaction design’, *First Monday*. Edited by N. Bonnie, K. Victor, and F. Kirsten. The MIT Press, 12(4), p. 333. doi: 10.5210/fm.v12i4.1772.

Karanasios, S., Riisla, K. and Simeonova, B. (2017) ‘Exploring the use of contradictions and tensions in activity theory studies: an interdisciplinary review’, in *33rd EGOS Colloquium: The Good Organisation*. Copenhagen: Loughborough University Institutional Repository, pp. 0–9. Available at: https://dspace.lboro.ac.uk/2134/26026.

Karlsson, F. and Wistrand, K. (2006) ‘Combining method engineering with activity theory: Theoretical grounding of the method component concept’, *European Journal of Information Systems*, 15(1), pp. 82–90. doi: 10.1057/palgrave.ejis.3000596.

Kassab, M. and Defranco, J. (2018) ‘An Empirical Investigation on the Satisfaction Levels with the Requirements Engineering Practices : Agile vs . Waterfall’, *2018 IEEE International Professional Communication Conference (ProComm)*. IEEE, pp. 118–124. doi: 10.1109/ProComm.2018.00033.

Kekwaletswe, R. and Gaoussou, A. (2012) ‘EFFECTIVE SKILLS TRANSFER THROUGH A LEARNING MANAGEMENT SYSTEM : A CASE OF ELECKOM CORPORATE SERVICES’, *IADIS International Journal on Computer Science and Information Systems*. Berlin, 7(2), pp. 57–68. Available at: http://www.iadisportal.org/ijcsis/papers/2012150105.pdf.

Kelly, A. (2017) *Kanban Paradox - DZone Agile*, *DZone Agile*. Available at: https://dzone.com/articles/kanban-paradox-1 (Accessed: 21 September 2018).

Khaled, E. E. and A. Gunes, K. (2004) ‘Applied Software Project Management’, *International Journal of Project Management*, 2(1), pp. 84–90. doi: 10.1177/109434200404853.

Kim, G. (2015) *Top 11 Things You Need to Know About DevOps*. Available at:

https://www.thinkhdi.com/~/media/HDICorp/Files/White-Papers/whtppr-1112-devops-kim.pdf (Accessed: 4 May 2017).

Kingdon, D. (2018) *Are Agile Principles More Important Than the Practices?*, *DZone*. Available at: https://dzone.com/articles/agile-principles-over-practices (Accessed: 20 September 2018).

Knauss, E. *et al.* (2017) 'Quality requirements in agile as a knowledge management problem: More than just-in-time', in *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017*, pp. 427–430. doi: 10.1109/REW.2017.35.

Koski, A. and Mikkonen, T. (2015) 'Requirements, architecture, and quality in a mission critical system: 12 lessons learned', *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*, pp. 1018–1021. doi: 10.1145/2786805.2804436.

Kroeze, J. H. (2012) 'Postmodernism, Interpretivism, and Formal Ontologies', in *Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems*. IGI Global, pp. 43–62. doi: 10.4018/978-1-4666-0179-6.ch003.

Kuhrmann, M. *et al.* (2016) 'How does software process improvement address global software engineering?', *Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016*, pp. 89–98. doi: 10.1109/ICGSE.2016.10.

Kumalo, N. and Poll, J. A. Van Der (2015) 'The role of cloud computing in addressing SME challenges in South Africa', *Proceedings - 2015 International Conference on Cloud Computing Research and Innovation, ICCCRI 2015*, pp. 139–147. doi: 10.1109/ICCCRI.2015.32.

Lampe, U. (2011) 'Optimizing the distribution of software services in infrastructure clouds', *Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011*, pp. 69–72. doi: 10.1109/SERVICES.2011.15.

Larman, C. and Basili, V. R. (2003) 'Iterative and incremental developments. a brief history', *Computer*, 36(6), pp. 47–56. doi: 10.1109/MC.2003.1204375.

Larrucea, X. *et al.* (2016) 'Software Process Improvement in Very Small Organizations', *IEEE Software*, 33(April), pp. 85–89. doi: 10.1109/MS.2016.42.

Lavallée, M. and Robillard, P. N. (2015) 'Why good developers write bad code: An observational case study of the impacts of organizational factors on software quality', *Proceedings - International Conference on Software Engineering*, 1, pp. 677–687. doi: 10.1109/ICSE.2015.83.

LeCompte, M. D. and Goetz, J. P. (1982) 'Problems of reliability and validity in Ethnographic Research', *Review of Educational Research*, 52(1), pp. 31–60. Available at: http://www.fctl.ucf.edu/researchandscholarship/sotl/creatingsotlprojects/implementingmanaging/content/LeCompteandGoetz.pdf.

Leffingwell, D. (2010) *Software Agile Requirements: lean requirements practices for teams, programs, and the enterprise*. Edited by Pearson. Boston: Addison Wesley.

Lethbridge, T. C. and Laganiere, R. (2004) 'Object-Oriented Software Engineering : Practical Software Development Using Uml and Java 2nd', *McGraw-Hill Publishing Company*. Available at: http://www.directtextbook.com/isbn/9780077109080.

Lima, A. M. (2010) 'Risk assessment on distributed software projects', *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, 2, p. 349. doi: 10.1145/1810295.1810387.

Lin, Y., Zhenyu, L. and Lizhi, C. (2012) 'Application and method of quality management for small-size software project', *Information Science and Control Engineering 2012 (ICISCE 2012), IET International Conference on*, pp. 1–5. doi: 10.1049/cp.2012.2347.

Linger, R. C. (1993) 'Clean room Software Engineering for Zero-Defect Software', in Alamitos, I. C. S. P. L. (ed.) *ICSE '93 Proceedings of the 15th international conference on Software Engineering*. Baltimore: IEEE, pp. 2–13. Available at: https://userweb.cs.txstate.edu/~rp31/papersSP/LingerCleanroom1993.pdf (Accessed: 27 December 2018).

Liu, J. *et al.* (2016) 'The Evaluation of the Embedded Software Quality Based on the Binary Code', *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 167–170. doi: 10.1109/QRS-C.2016.26.

Liu, X. Q., Kane, G. and Bambroo, M. (2006) 'An intelligent early warning system for software quality improvement and project management', *Journal of Systems and Software*, 79(11), pp. 1552–1564. doi: 10.1016/j.jss.2006.01.024.

Liyanarachchi, R. (2019) *Five Reasons Why Scrum Fails in Software Development - DZone Agile*, *Dzone Agile Zone*. Available at: https://dzone.com/articles/five-reasons-why-scrum-fails-in-software-developme?edition=457217&utm_source=Weekly Digest&utm_medium=email&utm_campaign=Weekly Digest 2019-03-06 (Accessed: 27 March 2019).

Logigear (2013) *How to Select the Right Tools for Testing in Agile ?*, *http://www.logigear.com*. Available at: http://www.logigear.com/blog/agile/using-tools-for-testing-in-agile/ (Accessed: 20 December 2017).

Longo, J. and Kelley, T. (2015) 'Use of GitHub as a Platform for Open Collaboration on Text Documents Justin', *Proceeding OpenSym '15 Proceedings of the 11th International Symposium on Open Collaboration*, 52, pp. 573–588. doi: 10.1016/j.chb.2015.01.050.

Lopez-Martinez, J. *et al.* (2016) 'Problems in the adoption of agile-scrum methodologies: A systematic literature review', in *Proceedings - 2016 4th International Conference in Software Engineering Research and Innovation, CONISOFT 2016*, pp. 141–148. doi: 10.1109/CONISOFT.2016.30.

Lucidchart (2017) *Complete Guide to Waterfall Project Management Methodology*, *https://www.lucidchart.com/blog*. Available at: https://www.lucidchart.com/blog/waterfall-project-management-methodology (Accessed: 6 October 2018).

Lytvynova, K. (2018) *Extreme Programming: Values, Principles, and Practices*, *DZone Agile*. Available at: https://dzone.com/articles/extreme-programming-values-principles-and-practice (Accessed: 23 September 2018).

Mahnic, V. and Drnovscek, S. (2005) 'Agile Software Project Management with Scrum', *Research Technology Management*, 59(1), pp. 21–29. Available at: https://www.researchgate.net/publication/228967959_Agile_Software_Project_Management_with_Scrum (Accessed: 15 June 2016).

Majowska, A. (2018) *Kanban Misconceptions and Myths*, *DZone Agile*. Available at: https://dzone.com/articles/kanban-misconceptions-and-myths-kanban-tool-blog (Accessed: 21 September 2018).

Mann, K. J., Murphy, T. E. and West, M. (2018) *Magic Quadrant for Enterprise Agile Planning Tools*. G00331978. Gartner. doi: G00331978.

Manning, J. (2017) 'In Vivo Coding', in *The International Encyclopedia of Communication Research Methods*. Hoboken, NJ, USA: John Wiley & Sons, Inc., pp. 1–2. doi: 10.1002/9781118901731.iecrm0270.

Martin, R. (2015) 'The Character and Competence to be a Leader in the 21st century: The General is still right. A Qualitative Theme Analysis of the Speech by General H. Norman Schwarzkopf, Jr. to the

U.S. Military Academy at West Point in 1991', p. 19. doi: 10.13140/RG.2.1.2780.4880.

Maskelyne, N. and Devant, D. (2013) *Our Magic The Art and Theory in Magic*. Edited by C. Bianchi. Magic Limited-Lloyd E. Jones.

Masombuka, T. and Mnkandla, E. (2018) 'A DevOps collaboration culture acceptance model', in *The Annual Conference of the South African Institute of Computer Scients and Information Technologists*, pp. 279–285. doi: 10.1145/3278681.3278714.

Mccall, J. A., Richards, P. K. and Walters, G. F. (1977) *Concept and Definitions of Software Quality*.

McDowell, C. *et al.* (2004) 'The effects of pair-programming on performance in an introductory programming course', *ACM SIGCSE Bulletin*, 34(1), p. 38. doi: 10.1145/563517.563353.

Mckenna, T. and Whitty, S. J. (2013) 'Agile is Not the End-Game of Project Management Methodologies', in *Annual Project Management Australia Conference Incorporating the PMI Australia National Conference (PMOz),*. Melbourne, pp. 17–18. Available at: https://eprints.usq.edu.au/23649/1/Agile is Not the End-Game of Project Management Methodologies.pdf?_escaped_fragment_= (Accessed: 24 March 2018).

Mckenzie, C., Tee, J. and Denman, J. (2013) 'Up, Up and Away: Java App Development Heads to the Cloud', *TheServerSide.com*, March, pp. 1–10. Available at: http://tngconsultores.com/kw/pluginfile.php/53/mod_glossary/attachment/1119/Java_eguide.pdf.

McManus, J. and Wood-Harper, T. (2007) 'Software engineering: A quality management perspective', *TQM Magazine*, 19(4), pp. 315–327. doi: 10.1108/09544780710756223.

Medeiros, J. *et al.* (2016) 'Towards a Model about Quality of Software Requirements Specification in Agile Projects', *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, (0), pp. 236–241. doi: 10.1109/QUATIC.2016.058.

Miller, F. (1998) *Aristotle's Political Theory*, *Stanford Encyclopedia of Philosophy*. Available at: https://plato.stanford.edu/entries/aristotle-politics/?mod=article_inline (Accessed: 7 September 2018).

Mnkandla, E. (2008) *A SELECTION FRAMEWORK FOR AGILE METHODOLOGY PRACTICES : A Family of Methodologies Approach*. University of the Witwatersrand.

Mnkandla, E. (2009) 'About software engineering frameworks and methodologies', *IEEE AFRICON Conference*, (0), pp. 1–7. doi: 10.1109/AFRCON.2009.5308117.

Mnkandla, E. (2012) *The impact of critical business data on organizations*, *African Journal of Business Management*. doi: 10.5897/AJBM11.1640.

Modi, R. (2017) *DevOps with Windows Server 2016*. Packt Publishing.

Monson, D. (2019) *Microservices Anti-Patterns - DZone Microservices*, *Dzone Microservice Zone*. Available at: https://dzone.com/articles/microservices-anti-patterns?utm_source=Top 5&utm_medium=email&utm_campaign=Top 5 2019-03-293 (Accessed: 30 March 2019).

Mora, M. *et al.* (2012) *Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems*. Edited by M. Mora et al. IGI Global. doi: 10.4018/978-1-4666-0179-6.

Mugeni, J. (2015a) *Ganizani Cloud Infrastructure Strategy Overview*. 1.0. Johannesburg.

Mugeni, J. (2015b) *Ganizani Strategy 2015 – 2020*.

Mwansa, G. (2015) *Exploring the development of a framework for agile methodologies to promote the adoption and use of cloud computing services in South Africa*. University of South Africa.

Mwanza-Simwami, D. (2001) 'Where Theory meets Practice : A Case for an Activity Theory based Methodology to guide Computer System Design Open Research Online The Open University ' s repository of research publications Where Theory meets Practice : A Case for an Activity Theory base', in *Eighth IFIP TC 13 Conference on Human-Computer Interaction*. Tokyo. Available at: https://www.researchgate.net/publication/42795931_Where_Theory_meets_Practice_A_Case_for_an_Activity_Theory_based_Methodology_to_guide_Computer_System_Design.

Nagar, P. and Thankachan, B. (2011) 'Software Reliability Engineering – A Review', 1(2), pp. 133–137.

Nakazawa, S. and Tanaka, T. (2016) 'Development and application of kanban tool visualizing the work in progress', *Proceedings - 2016 5th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2016*, pp. 908–913. doi: 10.1109/IIAI-AAI.2016.156.

National Planning Commission (2010) 'National Development Plan (2030)', *Department: The Presidency Republic of South Africa*, p. 70. doi: ISBN: 978-0-621-41180-5.

Di Nitto, E. *et al.* (2016) 'A software architecture framework for quality-aware DevOps', in *Proceedings of the 2nd International Workshop on Quality-Aware DevOps - QUDOS 2016*, pp. 12–17. doi: 10.1145/2945408.2945411.

Nixon, H. (2014) *Analyzing Qualitative Data - YouTube*. Youtube. Available at:

https://www.youtube.com/watch?v=BnDUARfEu5I (Accessed: 8 January 2019).

Noorman, M., Hussin, N. and Tarmuchi, N. (2008) 'An exploratory study on systems development methodologies for web-based applications', *Information Management & Computer Security*, 16(2), pp. 137–149. doi: 10.1108/09685220810879618.

Okpara, J. and Wynn, P. (2007) 'Determinants of small business growth constraints in a sub-Saharan African economy', *SAM Advanced Management Journal*, 72(2). doi: ISSN: 0036-0805.

Omran, A. (2008) 'AGILE CMMI from SMEs perspective', in *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*. Damascus: IEEE, pp. 1–8. doi: 10.1109/ICTTA.2008.4530352.

Orso, A. and Rothermel, G. (2014) 'Software testing: a research travelogue (2000–2014)', *Proceedings of the on Future of Software Engineering - FOSE 2014*, 2(1997), pp. 117–132. doi: 10.1145/2593882.2593885.

Oster, A. (2004) *Software Process Improvement Frameworks: Perceived Impact On the Quality Managment Practices And Financial Performance of Software Organizations*. Nova SouthEastern University. doi: UMI 3144708.

Othman, M. *et al.* (2010) 'A Review on Project Management and Issues Surrounding Dynamic Development Environment of ICT project: Formation of Research Area. Heterogeneous Data Mining Using Immune Network System View project Disaster Management View project A Review on Project Manag', *Article in International Journal of Digital Content Technology and its Applications*. doi: 10.4156/jdcta.vol4.issue1.10.

Owoseni, A. and Twinomurinzi, H. (2018) 'The dynamic capabilities of small and medium-scale enterprises using mobile apps in Lagos, Nigeria', *The Electronic Journal of Information Systems in Developing Countries*, (July), p. e12061. doi: 10.1002/isd2.12061.

Oxford, T. (2015) 'Big technology, small business', *Mail & Guardian*, 24 September. Available at: https://mg.co.za/article/2015-09-24-00-big-technology-small-business.

Öztürk, M. M., CİL, İ. and Zengin, A. (2015) 'Development of a Multi-Agent Framework for Software Quality', *ACM SIGSOFT Software Engineering Notes*, 40(1), pp. 1–10. doi: 10.1145/2693208.2693234.

Palaganas, E. C. *et al.* (2017) 'Reflexivity in Qualitative Research: A Journey of Learning', *The Qualitative Report*, 22(2), pp. 426–438. doi: December 2, 2017.

Palihawadana, S. *et al.* (2017) 'Tool support for traceability management of software artefacts with DevOps practices', *3rd International Moratuwa Engineering Research Conference, MERCon 2017*, pp. 129–134. doi: 10.1109/MERCon.2017.7980469.

Pan, W. F. *et al.* (2010) 'Measuring structural quality of object-oriented softwares via bug propagation analysis on weighted software networks', *Journal of Computer Science and Technology*, 25(6), pp. 1202–1213. doi: 10.1007/s11390-010-9399-9.

Parker, S. (2019) *Agile Project Management Explained – A Beginner's Guide - DZone Agile*, *Dzone Agile Zone*. Available at: https://dzone.com/articles/agile-project-management-explained-a-beginners-gui?edition=461193&utm_source=Weekly Digest&utm_medium=email&utm_campaign=Weekly Digest 2019-03-2 (Accessed: 27 March 2019).

Parzych, D. (2017) *Defining DevOps*, *DZone DevOps*. Available at: https://dzone.com/articles/defining-devops-1 (Accessed: 23 September 2018).

Patwardhan, A. *et al.* (2016) *Embracing Agile methodology during DevOps Developer Internship Program*, *arXiv.org*. Available at: http://arxiv.org/abs/1607.01893 (Accessed: 6 May 2018).

Perera, P., Bandara, M. and Perera, I. (2016) 'Evaluating the Impact of DevOps Practice in Sri Lankan Software Development Organizations', pp. 281–287. doi: 10.1109/ICTER.2016.7829932.

PMI and AgileAlliance (2017) 'Agile Practice Guide', *PMI Book*, pp. 1–168. Available at: https://www.pmi.org.

Poojary, N. (2015) *EC2 Container Service and Elastic Beanstalk: Docker on AWS*, *https://cloudacademy.com/blo*. Available at: https://cloudacademy.com/blog/amazon-ec2-container-service-docker-aws/ (Accessed: 10 February 2018).

Pospieszny, P. (2017) 'Software Estimation: Towards Prescriptive Analytics', *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, 6, pp. 221–226. doi: 10.1145/3143434.3143459.

Pressman, R. and Maxim, B. (2015) *Software Engineering: A Practitioner's Approach*. 8th edn. Edited by M. H. Education. New York: McGrawHill Higher Education. doi: 1 2 3 4 5 6 7 8 9 0 DOC/DOC 1 0 9 8 7 6 5 4.

PsySSA (2018) *Online Readings in Research Methods*. Psychological Society of South Africa. Available at: https://www.psyssa.com/wp-content/uploads/2018/10/ORIM-Chapter-2-Qualtitative-Data-Analysis_by-Archer-1.pdf (Accessed: 2 November 2018).

PWC (2016a) *Africa ' s CEO look to technology for growth*, *PWC Press Office*. Available at: http://www.pwc.co.za/en/press-room/africa-business-agenda.html (Accessed: 12 November 2016).

PWC (2016b) *Technology Sector Scorecard Q2 2016*. Available at: http://www.pwc.com/gx/en/technology/scorecard/assets/tech-scorecard-q2-2016.pdf.

PWC (2016c) *Technology Sector Scorecard Q3 2016*. Available at: http://www.pwc.com/gx/en/technology/scorecard/assets/tech-scorecard-q3-16.pdf.

Qumer, A. and Henderson-Sellers, B. (2008) 'An evaluation of the degree of agility in six agile methods and its applicability for method engineering', *Information and Software Technology*, 50(4), pp. 280–295. doi: 10.1016/j.infsof.2007.02.002.

Raam, G. (2017) *DevOps Mentality*, *DZone DevOps*. Available at: https://dzone.com/articles/devops-mentality (Accessed: 23 September 2018).

Radigan, D. (2017a) *Kanban*, *www.atlassian.com*. Available at: https://www.atlassian.com/agile/kanban (Accessed: 2 June 2017).

Radigan, D. (2017b) *Scrum A brief look into using the scrum framework for software development*, *www.atlassian.com*. Available at: https://www.atlassian.com/agile/scrum (Accessed: 3 May 2017).

Ralyte, J. *et al.* (2003) 'Towards a Generic Model for Situational Method Engineering', in *CAISE*. Velden, pp. 98–110. doi: 10.1007/3-540-45017- 3_9.

Reddy, G. (2009) 'Designing software project management models based on supply chain quality assurance practices', *2009 WRI World Congress on Computer Science and Information Engineering, CSIE 2009*. IEEE, 7, pp. 659–663. doi: 10.1109/CSIE.2009.953.

Regan, S. (2016) *What is ChatOps? A guide to its evolution, adoption, and significance*, *www.atlassian.com/blog*. Available at: https://www.atlassian.com/blog/software-teams/what-is-chatops-adoption-guide (Accessed: 1 June 2017).

Rehkopf, M. (2018) *Kanban vs Scrum*, *https://www.atlassian.com*. Available at: https://www.atlassian.com/agile/kanban/kanban-vs-scrum?utm_source=newsletter-email&utm_medium=email&utm_campaign=jira-insiders-newsletter_may-2018_EML-2779&jobid=103305803&subid=1251347578 (Accessed: 27 May 2018).

Reichert, A. (2016) *3 steps for customizing your agile processes*, *https://techbeacon.com*. Available at: https://techbeacon.com/app-dev-testing/3-steps-customizing-your-agile-processes (Accessed: 30

December 2018).

Rescher, N. (2003) *Epistemology, an Introduction to the theory of Knowledge*. SUNNY seri. Edited by G. J. Lucas. New York: State University of New York Press, Albany.

Ritchie, B. and Brindley, C. (2005) ‘ICT adoption by SMEs: Implications for relationships and management’, *New Technology, Work and Employment*, 20(3), pp. 205–217. doi: 10.1111/j.1468-005X.2005.00154.x.

Rochkind, M. J. (1975) ‘The Source Code Control System’, *IEEE Transactions on Software Engineering*, SE-1(4), pp. 364–370. doi: 10.1109/TSE.1975.6312866.

Runeson, P. and Höst, M. (2009) ‘Guidelines for conducting and reporting case study research in software engineering’, *Empirical Software Engineering*, 14(2), pp. 131–164. doi: 10.1007/s10664-008-9102-8.

Saldaña, J. (2009) *The Coding Manual for Qualitative Researchers. Sage Publication. London*. SAGE publications. doi: 10.1017/CBO9781107415324.004.

Salo, O. and Abrahamsson, P. (2008) ‘Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum’, *IET Software*, 2(1), p. 58. doi: 10.1049/iet-sen:20070038.

Samadhiya, D., Wang, S.-H. and Chen, D. (2010) ‘Quality models: Role and value in software engineering’, *International Conference on Software Technology and Engineering*, pp. 320–324. doi: 10.1109/ICSTE.2010.5608852.

Sarkan, H. M., Ahmad, T. P. S. and Bakar, A. A. (2011) ‘Using JIRA and redmine in requirement development for Agile methodology’, *2011 5th Malaysian Conference in Software Engineering, MySEC 2011*, pp. 408–413. doi: 10.1109/MySEC.2011.6140707.

Scacchi, W. (2001) *Process Models in Software Engineering*.

Scaled Agile, I. (2016) *Glossary | SAFe ® 4.0 for Lean Software and Systems Engineering*. Available at: http://www.scaledagileframework.com/glossary/.

Scheid, J. (2013) *CMMI Versus Agile: Which Is Best?, Bright Hub Project Management*. Available at: http://www.brighthubpm.com/agile/68474-cmmi-versus-agile-which-is-best/ (Accessed: 30 January 2016).

Schwartz, R. and Amaba, B. (2017) ‘Innovation program deployment for industries with irreversible processes', *2017 IEEE Technology and Engineering Management Society Conference, TEMSCON 2017*, pp. 427–431. doi: 10.1109/TEMSCON.2017.7998413.

Schwartzel, T. and Eloff, M. M. (2012) 'A Practical Approach to Theory Structuring and Analysis', in *Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems*. IGI Global, pp. 271–297. doi: 10.4018/978-1-4666-0179-6.ch014.

Sekaran, U. (2003) *Research Methods For Business*. Fourth Edi, *Research Methods For Business A Skill-Building Approach*. Fourth Edi. Edited by J. W. & Sons. Carbondale: Jon Wiley & Sons.

Sereda, G. (2019) *DevOps Is An Efficiency Booster - DZone DevOps*, *DevOps Zone*. Available at: https://dzone.com/articles/devops-is-an-efficiency-booster (Accessed: 24 April 2019).

Sharma, P. and Hasteer, N. (2016) 'Analysis of linear sequential and extreme programming development methodology for a gaming application', *International Conference on Communication and Signal Processing, ICCSP 2016*, pp. 1916–1920. doi: 10.1109/ICCSP.2016.7754505.

Shelly, G. and Rosenblatt, H. (2012) *System Analysis and Design*. 9th edn, *Shelly Cashman Series*. 9th edn. Edited by C. L. Course Technology. Botson: Course Technology, Cengage Learning.

Sidorva, A. *et al.* (2008) 'Uncovering the Intellectual Core of the Information Systems Discipline', *MIS Quaterly*, 32(3), pp. 467–482. Available at: http://www.jstor.org/stable/25148852%0D.

Siviy, J. and Penn, M. L. (2005) *Relationships Between CMMI ® and Six Sigma*.

Sommerville, I. (2010) *Software Engineering*. 9th edn, *Software Engineering*. 9th edn. Edited by M. Hirsch. Pearson Education as Addison-Wesley. doi: 10.1111/j.1365-2362.2005.01463.x.

De Souza, C. R. B. and Redmiles, D. (2003) 'Opportunities for extending activity theory for studying collaborative software development', *European Conference in CSCW (ECSCW 2003)*, pp. 14–18. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.132.2147&rep=rep1&type=pdf.

Sowunmi, O. Y. and Misra, S. (2015) 'An Empirical Evaluation of Software Quality Assurance Practices and Challenges in a Developing Country', *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 867–871. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.129.

Steward, D. (2007) *SOFTWARE CULTURE OF QUALITY : A SURVEY TO ASSESS PROGRESS IN*

IMPLEMENTING A CULTURE FAVORABLE FOR CONTINUOUS PROCESS IMPROVEMENT IN SOFTWARE DEVELOPMENT ORGANIZATIONS by Douglas E . Steward A Dissertation Presented in Partial Fulfillment Of the Requiremen. Capella University. doi: UMI 3259656.

Stillwell, M. and Coutinho, J. G. F. (2015) 'A DevOps approach to integration of software components in an EU research project', *1st International Workshop on Quality-Aware DevOps, QUDOS 2015 - Proceedings*, (c), pp. 1–6. doi: 10.1145/2804371.2804372.

Suhairi, I. *et al.* (2017) 'Analysing Qualitative Data Systematically using Thematic Analysis for Deodoriser Troubleshooting in Palm Oil Refining', 56(1998), pp. 1315–1320. doi: 10.3303/CET1756220.

Svorstøl, S.-O. (2017) *Tailoring Agile Methods for Large Projects A Case Study of a Large Agile Project*. Norwegian University of Science and Technology. Available at: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2471977/16904_FULLTEXT.pdf?sequence=1 .

Tan, P. S. and Yuan, X. M. (2005) 'ICT technologies for enabling software focused supply chains', *2005 3rd IEEE International Conference on Industrial Informatics, INDIN*, 2005, pp. 187–191. doi: 10.1109/INDIN.2005.1560374.

Šteinberga, L. and Šmite, D. (2011) *Towards Understanding of Software Engineer Motivation in Globally Distributed Projects*, *2011 IEEE Sixth International Conference on Global Software Engineering Workshop*. IEEE. doi: 10.1109/ICGSE-W.2011.31.

Theron, P. M. (2015) 'Coding and data analysis during qualitative empirical research in Practical Theology', *In die Skriflig/In Luce Verbi*, 49(3), pp. 1–9. doi: 10.4102/ids.v49i3.1880.

Thummadi, B. V., Shiv, O. and Lyytinen, K. (2011) 'Enacted routines in agile and waterfall processes', in *Proceedings - 2011 Agile Conference, Agile 2011*, pp. 67–76. doi: 10.1109/AGILE.2011.29.

Todorovic, A. (2015) *A beginner's guide to GitHub*, *opensource.com*. Available at: https://opensource.com/life/15/2/beginners-guide-github (Accessed: 5 May 2017).

Trauth, E. M. and Erickson, L. B. (2012) 'Philosophical Framing and Its Impact on Research', in *Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems*. IGI Global, pp. 1–17. doi: 10.4018/978-1-4666-0179-6.ch001.

Trendowicz, A. and Punter, T. (2003) 'Quality Modelling for Software Product Lines', *7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, QAOOSE 2003*, pp.

1–12.

Turner, D. (2017) *Word clouds and word frequency analysis in qualitative data | Quirkos qualitative analysis software*, *https://www.quirkos.com/blog*. Available at: https://www.quirkos.com/blog/post/word-clouds-and-word-frequency-analysis-in-qualitative-data (Accessed: 8 January 2018).

UNIDO (2016) *Industrial Development Report 2016: The Role of Technology and Innovation in Inclusive and Sustainable Industrial Development*.

Vaismoradi, M., Turunen, H. and Bondas, T. (2013) 'Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study', *Nursing and Health Sciences*, 15(3), pp. 398–405. doi: 10.1111/nhs.12048.

Varia, J. and Mathew, S. (2017) 'Overview of Amazon Web Services', *Amazon Web Services*, (January), p. 22. Available at: http://media.amazonwebservices.com/AWS_Overview.pdf.

Vasconcellos-Silva, P. R., Carvalho, D. and Lucena, C. (2013) 'Word frequency and content analysis approach to identify demand patterns in a virtual community of carriers of hepatitis C.', *Interactive journal of medical research*. JMIR Publications Inc., 2(2), p. e12. doi: 10.2196/ijmr.2384.

Ventureburn (2015) *Startup Survey*. Available at: http://ventureburn.com/2015/06/ventureburn-survey-sheds-light-on-sas-tough-but-pioneering-startup-industry/ (Accessed: 12 November 2016).

Villasana, G. and Castello, R. (2014) 'An agile software quality framework lacking', pp. 2–5. doi: 10.1109/WCCAIS.2014.6916549.

Villatore-Silva, T. (2012) *Creating an Enterprise-Wide Cloud Strategy*. WP-7166. Available at: https://www.scribd.com/document/193715670/NepApp-Creating-An-Enterprise-Wide-Cloud-Strategy.

De Villiers, M. R. (Ruth) (2012a) 'Models for Interpretive Information Systems Research, Part 1', in *Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems*. IGI Global, pp. 222–237. doi: 10.4018/978-1-4666-0179-6.ch011.

De Villiers, M. R. (Ruth) (2012b) 'Models for Interpretive Information Systems Research, Part 2', in *Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems*. IGI Global, pp. 238–255. doi: 10.4018/978-1-4666-0179-6.ch012.

Voas, J. and Kuhn, R. (2017) 'What Happened to Software Metrics?', *Computer*, 50(5), pp. 88–98. doi:

10.1109/MC.2017.144.

Wahaballa, A. *et al.* (2015) 'Toward unified DevOps model', *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, 2015-Novem, pp. 211–214. doi: 10.1109/ICSESS.2015.7339039.

Wallace, D. and Reeker, L. (2001) 'Software quality', in Abran, A. et al. (eds) *Guide to the software engineering book of knowledge*. Trial Vers. The Mitre Corp.

Walsham, G. (1995) 'The Emergence of Interpretivism in IS Research', *Information systems research: Issues, methods and practical guidelines*, 6(376–394). doi: 10.1287/isre.6.4.376.

Wei, P. *et al.* (2015) 'Visualization of police intelligence data based on word clouds', *Proceedings - 2014 10th International Conference on Computational Intelligence and Security, CIS 2014*. IEEE, pp. 539–543. doi: 10.1109/CIS.2014.145.

Wells, D. (2013) *Extreme Programming: A gentle introduction*, *extremeprogramming.org*. Available at: http://www.extremeprogramming.org/ (Accessed: 20 February 2016).

Welsh, E. (2002) 'Dealing with Data: Using NVivo in the Qualitative Data Analysis Process', *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*. Deutsche Forschungsgemeinschaft, 3(2). Available at: http://www.qualitative-research.net/index.php/fqs/article/view/865/1880&q=nvivo+manual&sa=x&ei=zah_t5pqoyubhqfe9swgbq&ved=0cc4qfjaj (Accessed: 29 December 2018).

Whittaker, J. A. and Voas, J. M. (2002) *50 Years of Software: Key Principles for Quality*. Available at: http://www2.southeastern.edu/Academics/Faculty/galkadi/420/notes/50years-quality.pdf (Accessed: 29 August 2018).

Wiggins, R. R. (1997) 'Sustaining competitive advantage: Temporal dynamics and the rarity of persistent superior economic performance', *Annual Meeting of the Academy of Management*, 5669(504), p. 30.

Williams, M. (2011) *PlayStation Network hack will cost Sony $170M*, *IDG News Service*. Available at: http://www.computerworld.com/article/2508315/computer-hardware/playstation-network-hack-will-cost-sony--170m.html (Accessed: 23 August 2013).

Willis, B. (2014) 'The Advantages and Limitations of Single Case Study Analysis', *Www.E-Ir.Info*, pp. 50–52. Available at: https://www.e-ir.info/2014/07/05/the-advantages-and-limitations-of-single-case-study-analysis/ (Accessed: 6 May 2018).

Wilson, W. M., Rosenberg, L. H. and Hyatt, L. E. (1997) ‘Automated analysis of requirement specifications’, *Proceedings - International Conference on Software Engineering*, pp. 161–171. doi: 10.1109/ICSE.1997.610237.

Wu, Y. *et al.* (2014) ‘Exploring the Ecosystem of Software Developers on GitHub and Other Platforms’, *CSCW conf.*, pp. 265–268. doi: 10.1145/2556420.2556483.

Xu, P. and Ramesh, B. (2008) ‘Using process tailoring to manage software development challenges’, *IT Professional*, 10(4), pp. 39–45. doi: 10.1109/MITP.2008.81.

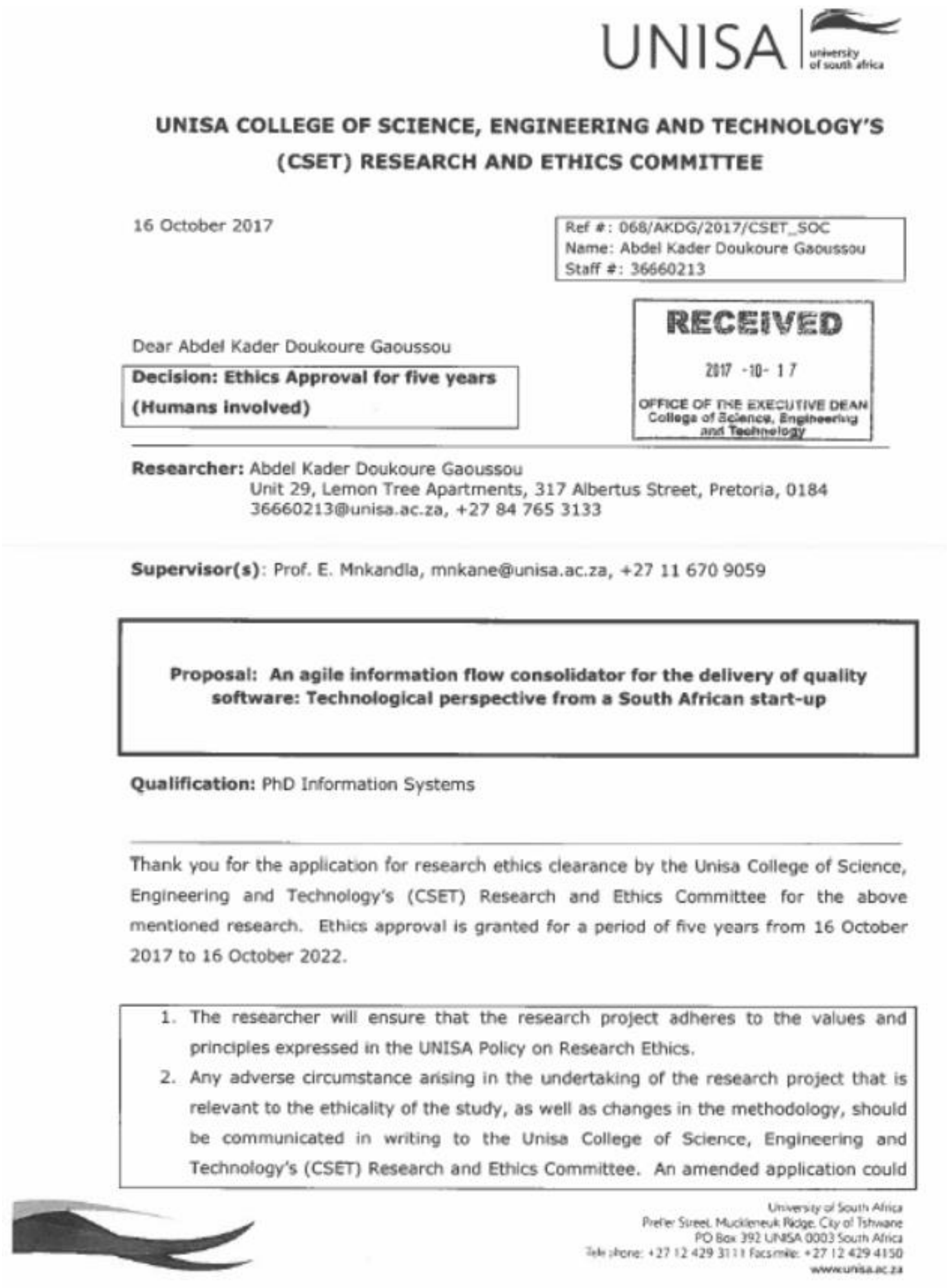
Yinfen, X. (2011) ‘The design of software process management and evaluation system in small and medium software enterprises’, *2011 International Conference on Computer Science and Service System (CSSS)*, pp. 2699–2701. doi: 10.1109/CSSS.2011.5974788.

Yoshida, T. (2018a) *A pretty good summary of Lean, Agile, Scrum*, *Medium*. Available at: https://medium.com/@takeshi.yoshida/a-pretty-good-summary-of-lean-agile-scrum-168cf123748 (Accessed: 17 December 2018).

Yoshida, T. (2018b) *Try Design Thinking + Scrum: A Powerful Hybrid Agile Approach*, *www.linkedin.com/pulse*. Available at: https://www.linkedin.com/pulse/try-design-thinking-scrum-powerful-hybrid-agile-approach-yoshida/ (Accessed: 17 December 2018).

Zahedi, M., Tessier, V. and Hawey, D. (2017) ‘Understanding Collaborative Design Through Activity Theory’, *The Design Journal*. Routledge, 20(sup1), pp. S4611–S4620. doi: 10.1080/14606925.2017.1352958.

Appendix A: Research ethics clearance certificate



UNISA university of south africa

UNISA COLLEGE OF SCIENCE, ENGINEERING AND TECHNOLOGY'S (CSET) RESEARCH AND ETHICS COMMITTEE

16 October 2017

Ref #: 068/AKDG/2017/CSET_SOC
Name: Abdel Kader Doukoure Gaoussou
Staff #: 36660213

RECEIVED
2017 -10- 17
OFFICE OF THE EXECUTIVE DEAN
College of Science, Engineering and Technology

Dear Abdel Kader Doukoure Gaoussou

Decision: Ethics Approval for five years (Humans involved)

Researcher: Abdel Kader Doukoure Gaoussou
Unit 29, Lemon Tree Apartments, 317 Albertus Street, Pretoria, 0184
36660213@unisa.ac.za, +27 84 765 3133

Supervisor(s): Prof. E. Mnkandla, mnkane@unisa.ac.za, +27 11 670 9059

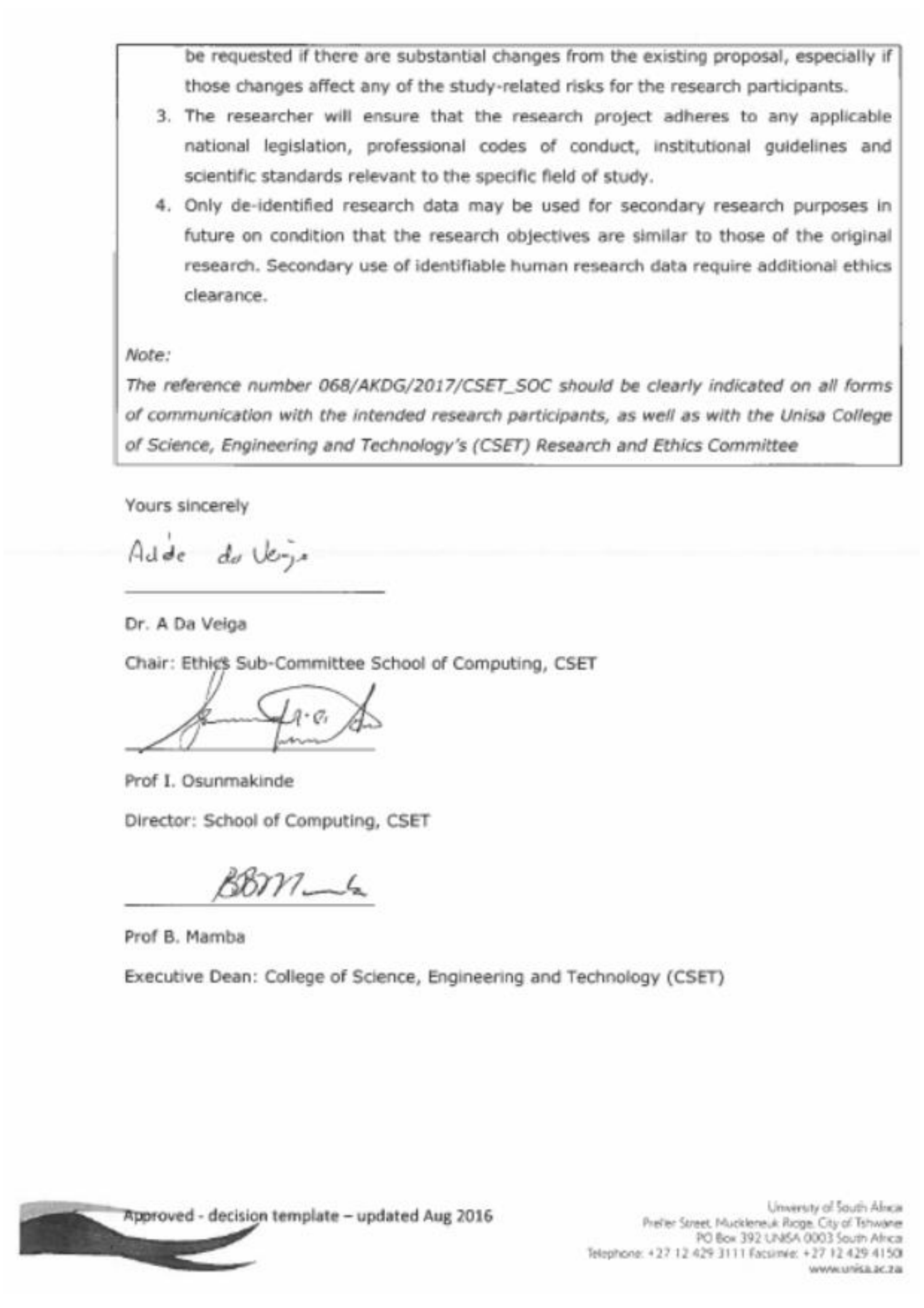
Proposal: An agile information flow consolidator for the delivery of quality software: Technological perspective from a South African start-up

Qualification: PhD Information Systems

Thank you for the application for research ethics clearance by the Unisa College of Science, Engineering and Technology's (CSET) Research and Ethics Committee for the above mentioned research. Ethics approval is granted for a period of five years from 16 October 2017 to 16 October 2022.

1. The researcher will ensure that the research project adheres to the values and principles expressed in the UNISA Policy on Research Ethics.
2. Any adverse circumstance arising in the undertaking of the research project that is relevant to the ethicality of the study, as well as changes in the methodology, should be communicated in writing to the Unisa College of Science, Engineering and Technology's (CSET) Research and Ethics Committee. An amended application could

University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za



be requested if there are substantial changes from the existing proposal, especially if those changes affect any of the study-related risks for the research participants.

3. The researcher will ensure that the research project adheres to any applicable national legislation, professional codes of conduct, institutional guidelines and scientific standards relevant to the specific field of study.
4. Only de-identified research data may be used for secondary research purposes in future on condition that the research objectives are similar to those of the original research. Secondary use of identifiable human research data require additional ethics clearance.

Note:

The reference number 068/AKDG/2017/CSET_SOC should be clearly indicated on all forms of communication with the intended research participants, as well as with the Unisa College of Science, Engineering and Technology's (CSET) Research and Ethics Committee

Yours sincerely

[illegible]

Dr. A Da Veiga

Chair: Ethics Sub-Committee School of Computing, CSET

[illegible]

Prof I. Osunmakinde

Director: School of Computing, CSET

[illegible]

Prof B. Mamba

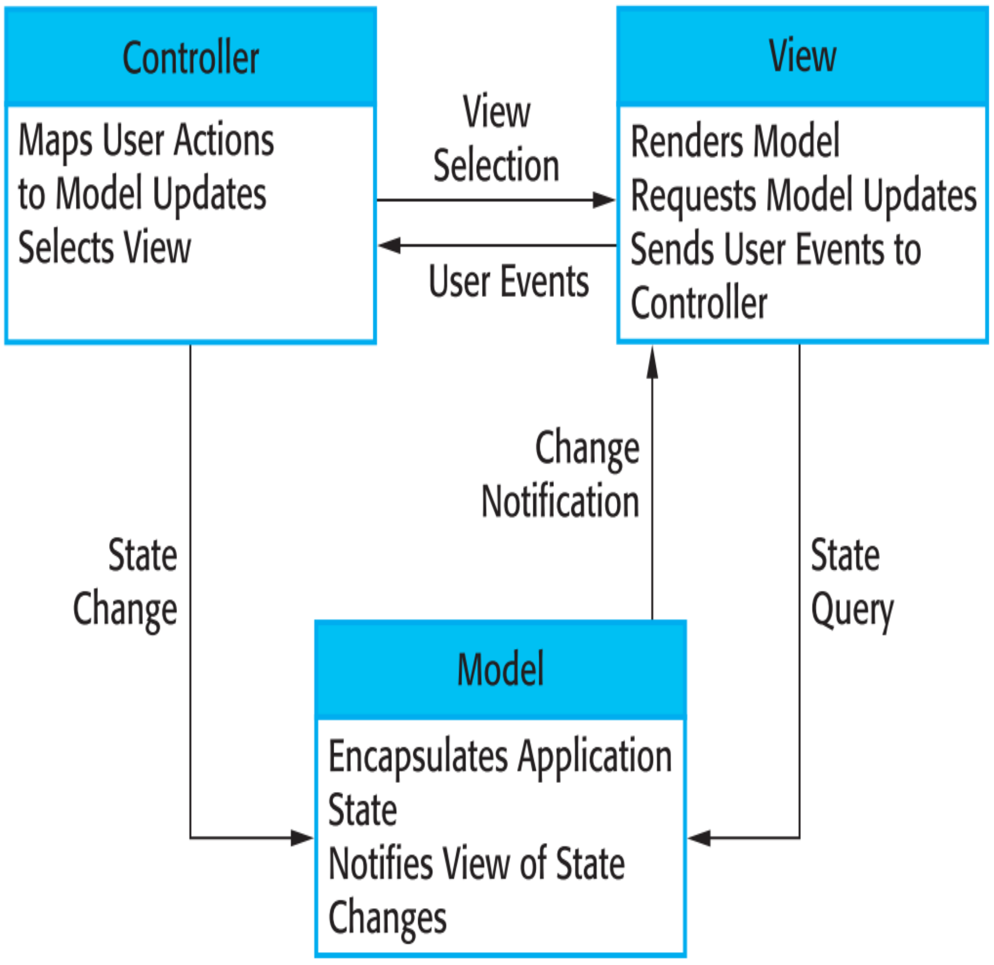
Executive Dean: College of Science, Engineering and Technology (CSET)

Appendix B: Model-View-Controller Pattern (MVC) OUTLINE

Model-View-Controller (MVC) is an architectural pattern which essentially advocates segregation of the presentation layer from business layers and backend data. It aims to separate user interfaces from other parts of the system by loosely coupling the presentation layer and functional components (Lethbridge and Laganiere, 2004, p. 355). In a web architectural scenario, MVC divides three main system components as follows:

MVC COMPONENT	**DESCRIPTION**
Model	*The model refers to all components and classes which contain underlying business objects and data.*
View	*The view refers to all the components which will translate business data into a digestible presentation layer. It caters for the channel clients use to access the system and basically determines how the presentation layer will be rendered as a user interface.*
Controller	*This layer contains the component which will integrate requests from the view and renders responses from the model layer.*

The MVC pattern is often used in scenarios when one wants to cater to different channels for the viewing or manipulation of information from an end-user perspective. Additionally, when one is looking for flexibility and the ability to cater for future needs, it permits modification and upgrades to presentation or model layers without having an incurrence on these components (Sommerville 2010, p. 155):


Controller
Maps User Actions to Model Updates
Selects View
View Selection
User Events
View
Renders Model
Requests Model Updates
Sends User Events to Controller
Change Notification
State Change
State Query
Model
Encapsulates Application State
Notifies View of State Changes

Appendix C: Architecture, DevOps tooling and related terms

A more detailed list can be viewed at http://aspetraining.com/files/DevOps-Tools-glossary.pdf

- **Amazon Web Services (AWS)**: Cloud services leader in the IAAS, PAAS, SAAS space
- **Glassfish**: Open-Source application server considered the JEE reference for running web-based application and services
- **Jenkins**: Leading open-source continuous integration server
- **JIRA**: Atlassian tool used for issue tracking as well as overall agile project management support
- **GITHUB:** Socially driven code repository built on top of the GIT SCM scheme
- **TOGAF:** Enterprise Architecture framework from the Open group. See https://www.opengroup.org/togaf

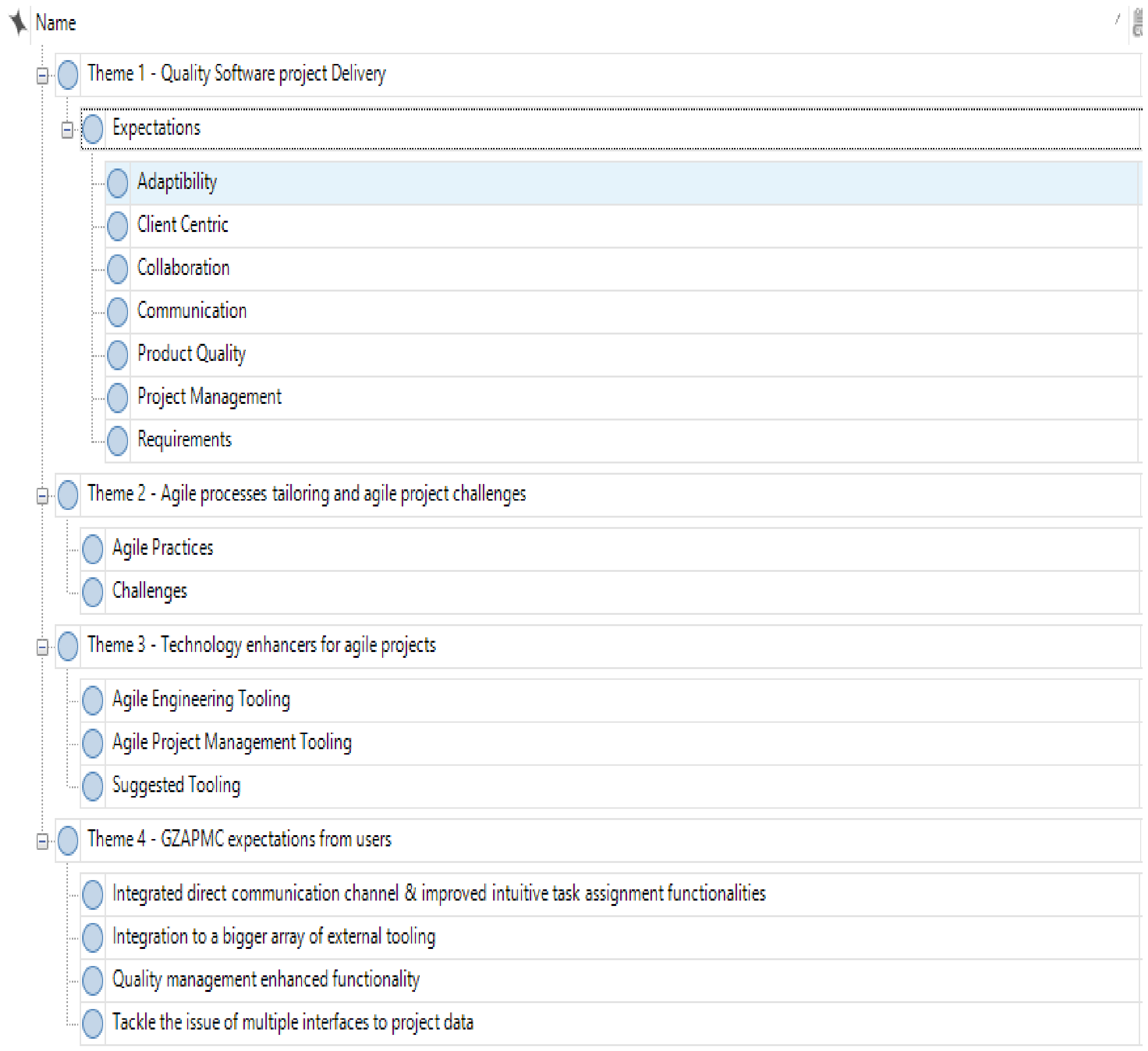
Appendix D: Literature review underpinning themes

In conducting the thematic analysis for literature review, various concepts were discussed. Underpinned by stated research objectives in Chapter 1, the themes below were explored in the literature survey:

- Research Objective 1: Implications of quality software projects in modern settings
    - Discussion on how the software crisis led to the need for software engineering.
    - Concepts of quality in software, its dimensions and business values.
    - Quality management and models.
    - Agile approaches to software development projects.
- Research Objective 2: Discuss the importance of tailoring development process to organisational and market specific needs.
    - Concepts of software and system process engineering.
    - Method engineering.
    - Discussion on the importance to customise agile processes and contextualisation to the SMEs environments in sub-Saharan markets.
- Research Objective 3: Discuss the technology tooling available to facilitate agile development and the delivery of quality software solutions to customers.
    - The emergence of cloud computing.
    - Source Control in the cloud: GITHUB.
    - Infrastructure as Service: Amazon Web Services.
    - Agile Project Management: The Atlassian suite.
    - Continuous Integration: Jenkins.
    - ChatOps: Slack
    - Discussion on the limitations to siloed usage of agile and Devops related tooling.
- Research Objective 4: Develop an information system to complement existing tools, facilitate the management of agile development projects and facilitate quality software products delivery

Appendix E: Semi-structured interview underpinning themes

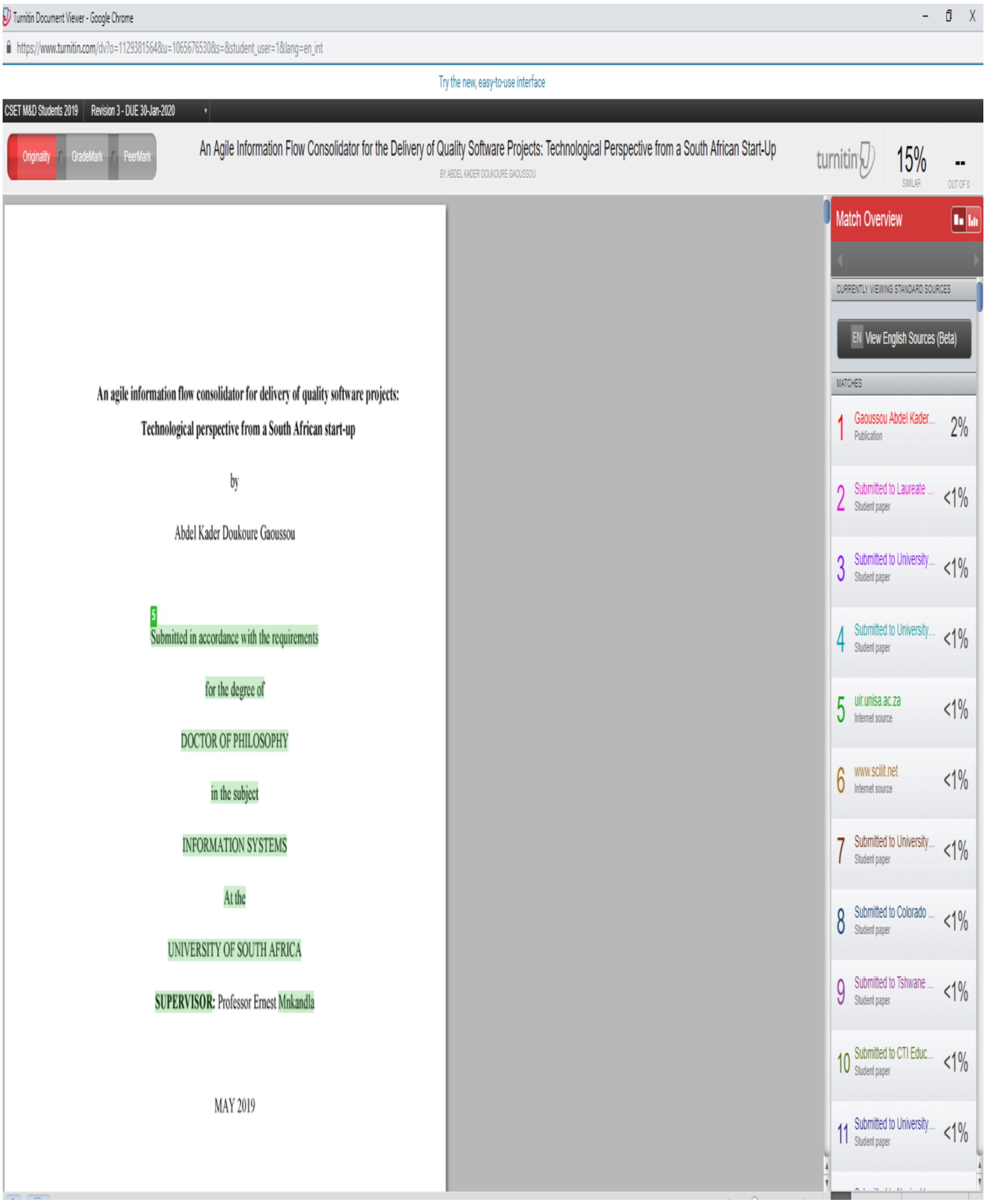
To produce a themeing data analysis based on identified anchor codes for the interviewing process, various concepts and meta-themes were derived. The underpinning themes below where the main drivers for these semi-structure interviews with engineering and management staff:



Name

- Theme 1 - Quality Software project Delivery
  - Expectations
    - Adaptibility
    - Client Centric
    - Collaboration
    - Communication
    - Product Quality
    - Project Management
    - Requirements
- Theme 2 - Agile processes tailoring and agile project challenges
  - Agile Practices
  - Challenges
- Theme 3 - Technology enhancers for agile projects
  - Agile Engineering Tooling
  - Agile Project Management Tooling
  - Suggested Tooling
- Theme 4 - GZAPMC expectations from users
  - Integrated direct communication channel & improved intuitive task assignment functionalities
  - Integration to a bigger array of external tooling
  - Quality management enhanced functionality
  - Tackle the issue of multiple interfaces to project data

Appendix F: Turnitin report


Turnitin Document Viewer - Google Chrome

https://www.turnitin.com/dv?o=1129381564&u=1065676530&s=&student_user=1&lang=en_int

Try the new, easy-to-use interface

CSET M&D Students 2019 | Revision 3 - DUE 30-Jan-2020

Originality | GradeMark | PeerMark

An Agile Information Flow Consolidator for the Delivery of Quality Software Projects: Technological Perspective from a South African Start-Up

BY ABDEL KADER DOUKOURE GAOUSSOU

turnitin 15% SIMILAR -- OUT OF 0

An agile information flow consolidator for delivery of quality software projects:

Technological perspective from a South African start-up

by

Abdel Kader Doukoure Gaoussou

5
Submitted in accordance with the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in the subject

INFORMATION SYSTEMS

At the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: Professor Ernest Mkandla

MAY 2019

Match Overview

CURRENTLY VIEWING STANDARD SOURCES

EN View English Sources (Beta)

MATCHES

1	Gaoussou Abdel Kader... Publication	2%
2	Submitted to Laureate ... Student paper	<1%
3	Submitted to University... Student paper	<1%
4	Submitted to University... Student paper	<1%
5	uir.unisa.ac.za Internet source	<1%
6	www.scilit.net Internet source	<1%
7	Submitted to University... Student paper	<1%
8	Submitted to Colorado ... Student paper	<1%
9	Submitted to Tshwane ... Student paper	<1%
10	Submitted to CTI Educ... Student paper	<1%
11	Submitted to University... Student paper	<1%